THE BEST SELLING COMPUTER PROJECTS MAGAZINE AUGUST 1985

ELECTRONICS & COMPUTING

ORIC TAKEOVER 9
10
14
31
45
48
PLOTTER 52
MODEMS LIST 54
57
58
65 PLOTTER IC in

AN EMAP PUBLICATION

USA $2.95
Germany D6.00
Singapore S$4.95
95p

★ BBC ★ SPECIAL

DIY 6809 SECOND PROCESSOR

PLUS

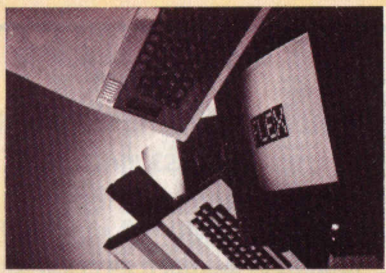
ROM DESIGNER

MAKING THE MOST OF WORDWISE

WHICH MODEM? – ACROSS THE BAUD SURVEY

REMEDIES FOR A SICK DISK DRIVE

QL SOFTWARE - AMSTRAD ASSEMBLERS - SANYO 555



Special Offer

BBC 'experimenter' ROM

See page 13 for details

Electronics & Computing Monthly
Priory Court, 30-32 Farringdon Lane,
London, EC1R 3AU

Editorial 01-251-6222
Editor Gary Evans
Deputy Editor William Owen

Advertising 01-251-6222
Advertisement Manager Tony Herman
Advertising Executive Tracey Keighley
Advertising Production Serena Hadley

Production 01-251-6222
Design Boldface
Typesetting & Make-up Time Graphics

Publisher Terry Pratt

Distribution
EMAP National Publications

Published by
EMAP Business and
Computer Publications

Printed by
Riverside Press, England

Subscriptions and Back Issues
Subscriptions/
Back Issues telephone 0858 34567

Electronics & Computing Monthly is normally published on the 13th day of each month.

ELECTRONICS & COMPUTING

Contents

Vol. 5 Issue 8

SOFTWARE

PROJECTS

FEATURES

REVIEWS

PLUS

Further news on some exciting changes to the editorial style and content of Electronics and Computing can be found on page 22.

NEWS NEWS NEWS NEWS

No Amstrad 64K upgrades available for 32K users

Our review of the Amstrad CPC664 in the June issue mentioned the fact that company chairman Alan Sugar had said that sympathetic consideration would be given to any 464 owner who wished to upgrade the machine by installing a 664 ROM. It now seems that the sympathetic ear has become deaf.

Company spokesman Roland Perry said that Mr Sugar's statement had been made in the heat of the moment at the press launch of the computer. He subsequently decided to change his mind, yet to our knowledge did not communicate this change of policy to the press.

The reasons for the change of heart, as given by Mr Perry, are rather 'high handed', his words not ours. It seems that Amstrad wish to protect the public from themselves. Even if the person wishing to upgrade a 464 to a 664 is aware of the fact that some 'non-standard' software will not run on the modified computer, Amstrad will not supply the new ROM.

The only people that Amstrad is prepared to supply the new ROM to are those in the trade who wish to write V1.1 software. So the ROM is available but not to the great unwashed, in which category, unfortunately, *E&CM* readers must see themselves.

. . . and meanwhile

Amstrad chose the occasion of the Chicago Consumer Electronics Show to launch its assault on the American market. The machine on which all hopes are pinned is a version of the CPC664 fitted with 128K of onboard RAM. The 6128 is to sell for £699 if bundled with a green screen monitor or £799 with RGB monitor.

Amstrad Chairman Alan Sugar is quoted as saying that the company has no plans to sell the 128K machine in the UK but we've all heard this sort of denial in the past. Smart money has it that the machine will make a UK appearance in the autumn, some reports even claim that a ship carrying a consignment of 6128s has left the Far East destined for these shores.

The question facing people considering the purchase of a 664 is whether or not to wait for the new computer. The answer must be to wait if the machine is to be used to run applications software under CP/M. The 64K RAM of the 664 is a limitation when attempting to run CP/M software – full details in a special feature on the 664 in the near future.

Robots fail to arouse interest

While the home computer industry is going through one of the toughest periods in its short history – witness the never ending series of company collapses/financial rescue plans, the personal robotics industry seems to have been still born. Over a year ago we at *E&CM* attempted to stimulate growth in this area by increasing our coverage of small, low cost robotic systems via our **'Your Robot'** supplement. The interest shown by both readers and advertisers with products to sell was however, to put it politely, negligable.

At the time many people were predicting that the market for robots was about to 'take off' in the way that microcomputers had a few years ago. But despite considerable enthusiasm in many quarters, the market for home robots has failed to materialise. The problems in the computer industry have contributed to a dampening of interest in robotics and the related subject of computer control systems.

One company that can testify to the indifferent climate is Oyez Scientific and Technical Services Ltd. It had planned to stage a three day conference and exhibition at which speakers would present a series of talks tracing the history of small scale robot systems and providing some thoughts on the future shape of the industry. In order to publicise the conference some 25,000 leaflets were mailed out to likely delegates. In the event, the massive mail out generated only a handful of firm bookings. In a rapid rethink, Oyez then decided to restrict the conference to a single day and reduced the registration fee from the original £250 to £75.

It remains to be seen whether this scaled down version of the original conference will attract any degree of support: we'll report on that next month. But Oyez' current experience confirms the lack of support for home robotics in this country.

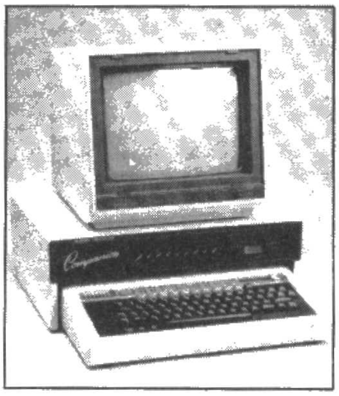
What is even more depressing is that many leading figures in the field of small robotic systems privately concede that the demand for their products is minimal.

GARY EVANS

Visit E&CM at Acorn show

The third Acorn User Exhibition will be staged at the Barbican Centre between the 25th and 28th of July and *Electronics and Computing* will be there. We'll have a number of projects on display and will be giving away prizes throughout the show. Editorial staff will be on hand to answer any burning questions you want to ask or just to chat about the current state of the computer market.

Make a date in your diary and note that our stand number is 118.



Monitor stand couples as disk drive

Bevan Technology's Companion is a multi-purpose add-on for the BBC micro. On a mundane level it provides a bridge over the computer allowing a monitor to be used in a convenient position. Built into the unit is a 3½" disk drive and circuitry that allows the micro monitor and control external hardware.

Full technical details from Bevan at Gresham Chambers, 14 Lichfield Street, Wolverhampton, West Midlands, WV1 1DG. Telephone 0902 23546 or 23746.

How to build your own PC and other tales

Readers of Byte will be familiar with the name Steve Ciarcia but for those not in the know, Mr. Ciarcia is the author of a regular feature 'Ciarcia's Circuit Cellar' which presents projects ranging from 'How to build your own PC' to more mundane topics such as the construction of an RS232 breakout box.

McGraw-Hill has published the latest in a series of books that collect together the more popular of the Byte projects.

'Ciarcia's Circuit Cellar – Volume 4' should now be available for £15.95 at your local book store. The price may seem high but for anyone with an interest in the latest circuit techniques the book represents excellent value for money.

Oric revived en France

The announcement that a French company, Eureka Informatique, has purchased Oric Products International will come as no surprise to watchers of the microcomputer industry.

The Oric computer and its derivatives, the Atmos and Stratos, never managed to achieve a significant penetration of the UK market but had a moderate degree of success across the Channel. Eureka Informatique is an established distributor of micro hardware, including products from Sinclair, Amstrad and Enterprise, and sees the Oric as a machine that will complement their existing range of products. Eureka has 'an interest' in a manufacturing facility in Normandy and plans to continue production of Oric products.

UK owners of Oric computers will be able to contact the French company for service and parts in future but is should be noted that any warranties in existence on Oric computers lapsed when the original company went into receivership.

This picture of Sir Clive Sindair was taken just days before his company was taken over by Mr. Robert Maxwell. No wonder he's finding it difficult to concentrate on the Penman Plotter that's just been shoved under his nose; he obviously had weightier things on his mind.

Those little perforations

Removing the perforated strips from the edges of computer printout paper, while simple in theory, often results in an annoying tear in the printed area of the copy. This problem is particularly evident when removing the edges from single sheets of paper.

Easystrip is a new device intended to overcome these problems. It can be used on both perforated and non-perforated paper and in use is clamped over the margin of the paper, held in place by a series of pegs. A cutter blade then cleanly slices off the edges of the paper.

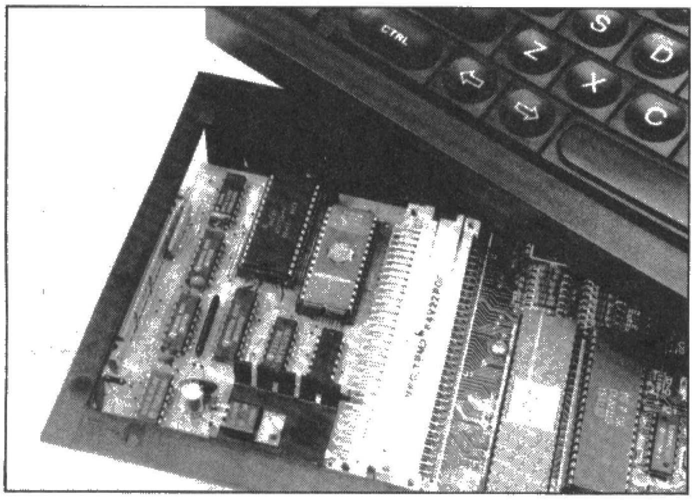
Further information from Associated Computer Supplies, Bowmaker House, Etruria Road, Hanley, STOKE-ON-TRENT, ST1 5NH.

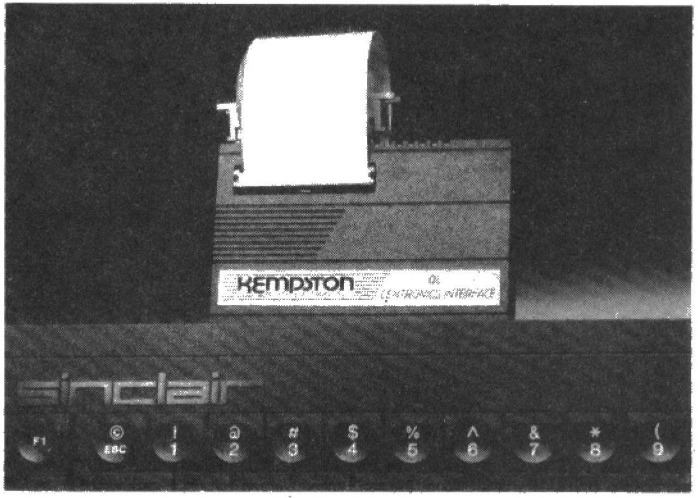
Acorn falls again

As we close for press the news that Acorn shares have once again been suspended has hit the headlines. It is obvious that Acorn's products are failing to sell and that cash flow is continuing to be a problem.

As far back as the 1st of May, Acorn's desperate need for cash was highlighted by a letter from one of its marketing consultants. This offered magazines the chance of buying a BBC B+ at a 'special price' for the purpose of review. To add insult to injury, the special price was only £50 below that of the Amstrad 664's recommended retail price and for that money you get a green screen monitor and CP/M thrown in.

Needless to say, Acorn's offer was one we felt we could refuse. But then perhaps the offer was not aimed at computer magazines for although the letter outlining the scheme bore the name of *E&CM's* Editor, it was addressed to the offices of 'Smash Hits', one of the many other magazines published by EMAP, publishers of *E&CM.*





Double helping of QL products from Kempston

Kempston Micro Electronics has added two QL products to its existing range of peripherals. A disk interface provides all the built in microdrive filing system operations but also includes the extensions that form part of the Sinclair QL Toolkit. In addition an 8K on-board ROM contains all the Toolkit file handling and job control utilities.

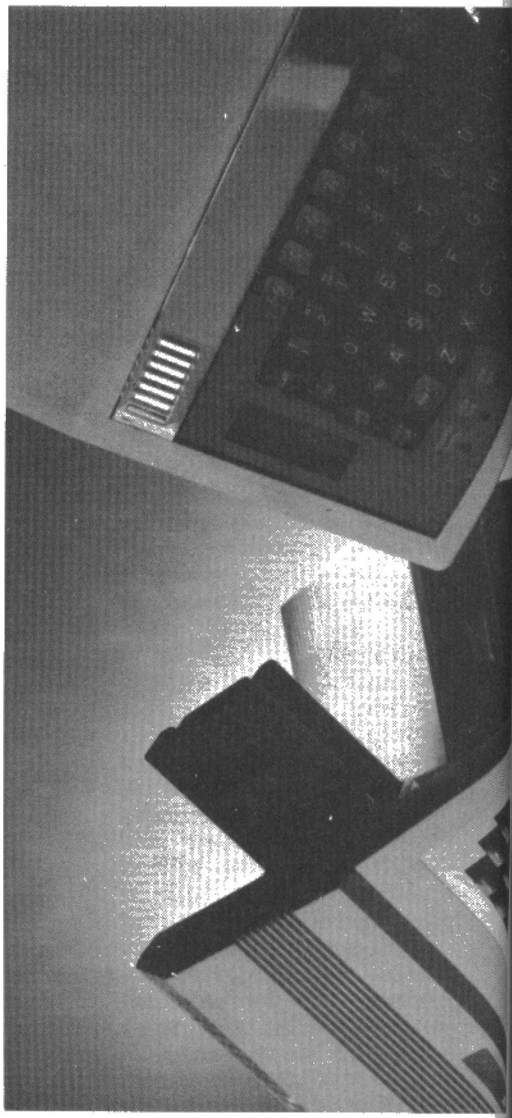
The £115 interface supports independently powered 3, 3½ or 5¼" drives.

Kempston's Centronics Interface is unusual in that it plugs into the ROM cartridge slot rather than either of the serial ports. It provides a number of advantages over other interfaces available. These include an efficient arrangement of buffers and the ability to store numerous print files without tying up the port.

Driving software is ROM resident and supports screen dumps in nine different formats including colour.

The Centronics interface costs £39.95 and both units are available direct from Kempston at 1-3 Singer Way, Kempston, Bedford. Telephone 0234 856633.

A DRAGON IN THE TUBE



Yes, a Dragon can be used as a 6809 second processor for the BBC micro. Hew Jones' simple interface opens up wider applications, a better operating system, and more memory to the BBC user.

The BBC micro, while undoubtedly a very powerful computer, has a number of serious drawbacks. Apart from the obvious problems caused by the chronic shortage of memory when using the high resolution screens, Acorn's DFS does not bear close scrutiny when compared to systems running a proprietary DOS such as CP/M. Such operating systems can boast a wealth of development software such as high level languages and cross assemblers. Utilities written for the BBC micro often try their best to 'make do' but rarely succeed in overcoming the lack of available memory (BCPL is a case in point).

Having highlighted the failings of the beeb, what are the positive aspects of the computer's design? Unquestionably, it is a superb example of digital design. The operating system software is excellent too, but there is rather a lot of it – 32K of ROM just to run BASIC is a bit excessive. A major bonus is that all of the interfaces found as optional extras on many home computers are fitted as standard on the beeb, or can simply be installed by adding a few components to the computer's main PCB. These interfaces include the floppy disk interface controller, printer port, RS423 port, A to D convertors and no less than three system expansion ports: the 1MHz bus, the user port and the TUBE. This list excludes the obvious attractions of an 80 x 24 text display which is available from any of three video outputs.

A second processor complements the BBC micro by taking over all the computation heavy workload leaving the BBC to concentrate on implementing the I/O. While there are many commercial second processors available, including Acorn's own 6502 and Z80 designs, this project makes use of a Dragon computer to offer a 6809 second processor system.

Enough has been said in praise of the 6809 MPU and it is a matter of regret that the Dragon never achieved the commercial success that it deserved. The designers of the computer did themselves no favours by restricting the display to a 32 x 16 upper case text only but they did build in 64K of RAM and a bus expansion facility. This is exactly the specification expected of a 6809 second processor.

Accepting that we are to use a Dragon computer as a second processor, what do we do with it? A simple approach would be to use the BBC micro as an extension of the Dragon's I/O capability. This could be accomplished by attaching appropriate software onto the Microsoft BASIC 'hooks', most existing Dragon software could be made to operate in a smart 80 column mode with disks. Most people would however, not unreasonably, expect a 6809 second processor to run a 6809 operating system – either OS9 or FLEX. The former is an excellent package supporting multi-tasking (great!) and multi-users (impractical in a floppy disk based system). Unfortunately, with the exception of Dragon Data's version, OS9 is very expensive. Conversely, FLEX has a broader market and is generally lower in cost with much more software in existence. It is well documented, available in a wide variety of standard disk formats, simple to use and convenient for development work. Both operating systems are capable of being customised for new hardware configurations, the FLEX Programmer's Manual being very helpful in this respect. All things considered, it had to be FLEX. A final nail in the coffin of OS9 is the fact that the Dragon version is supplied on double density disks which cannot be read by the 8271 FDC in the beeb.

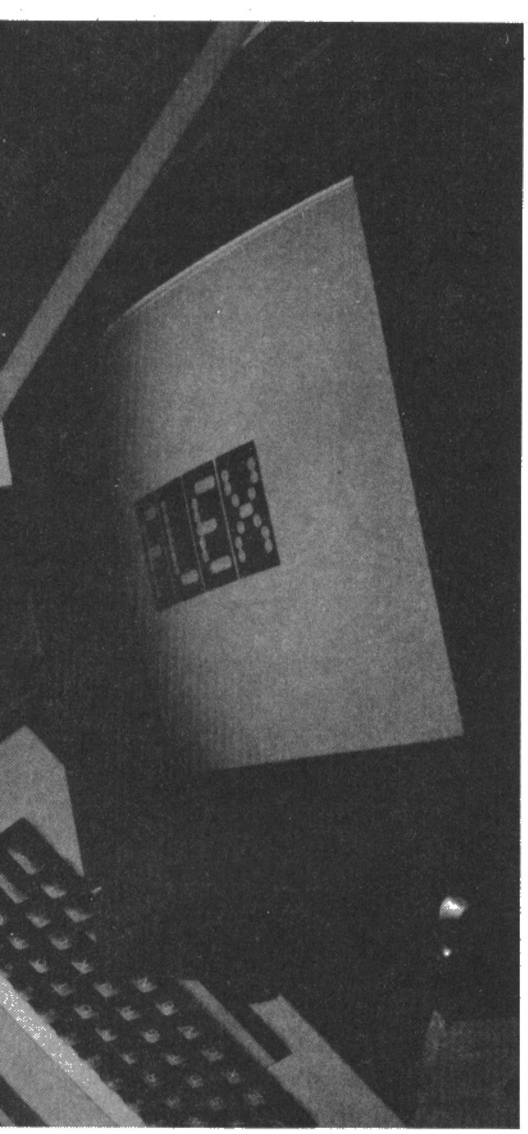
For those who possess a BBC model B with Acorn's DFS and dual disk drives then adding this second processor will not incur much expense apart from the purchase of the FLEX licence, ie a copy of the DOS. Ignoring the low component cost of the TUBE interface itself, the only other item required is the Dragon computer. A Dragon 64 is ideal but a Dragon 32 can be used providing that it is fitted with 64K of addressable RAM.

THE FLEX OS

The FLEX operating system originated in the United States. Written by TSC, FLEX is a powerful OS designed make the best use of the features of the 6809 MPU.

A wide range of applications packages is available for FLEX systems. These range from software development aids to a full suit of general and business software. Such packages include a variety of word processors; Dynacalc, a powerful spreadsheet system; the RMS data management system; and specialised business software such as cash and VAT and purchase ledger packages.

Compusense offers full support for FLEX in the UK and via its contacts in the USA has access to a vast number of applications packages. *Compusense, PO Box 169, 286D Green Lanes, Palmers Green, LONDON, N13 5XA.*



How it works

Despite all the mysterious references to the TUBE in the BBC User Guide, it is merely an extension of the CPU bus, brought 'raw' to the 40 way IDC TUBE plug. A TUBE select signal is provided which becomes active in the range $FEE0-$FEFF. External hardware is required before the TUBE can do anything at all and even then any data transfer must rely heavily on software. Acorn's answer was to produce a custom ULA to interface the micro to its co-processor whilst other designs have used the simple expedient of using two back-to-back VIAs. This design goes one better and implements a 2MHz TUBE interface with just a single 8255 PPI and a few support TTL chips.

The complete circuit diagram of the TUBE interface is shown in **Figure 1.** Connector CONN 1 is a 40 way IDC header which mates with one end of a 40 way ribbon cable terminated at both ends by 40 way IDC sockets. The other end plugs into the TUBE connector of the BBC micro. Keep the cable short, say 2ft maximum. CONN 2 is a 0.1" pitch double side edge connector which plugs into the Dragon cartridge port.

IC4 is an LS245 bi-directional buffer which isolates the 8255 from the 6502 data bus until the TUBE select goes low (SHEILA, offset E0-FF). Similarly IC5, an 81LS95, buffers the 6502 control signals. The outputs from this chip, and hence IC4, are only enabled when both the BBC and the Dragon are powered up. If the Dragon is on but the BBC off then IC5 and IC4 have no supply rail whilst if conditions are reversed, Q2 will be off and its collector will pull pin 19 of IC5 high. In this tri-state condition, IC4 is also disabled due to R2 pulling pin 19 high.

The correct power-up sequence is to first switch on the BBC micro, followed by the Dragon when the beeb has made its customary 'ready' tone. As part of the BBC computer's initialisation procedures, the computer's MOS polls for the presence of a second processor and will perform a TUBE set-up routine if it decides that there is one. This assumes Acorn's own ULA to be fitted. In the absence of formal documentation and the lack of inclination to disassemble the BBC ROM, it has been presumed that the BBC micro acknowledges the presence of a second processor if, and only if, it reads a $FF at $FEE0 immediately after a hard reset. Until its mode of operation is configured by the BBC, the 8255 will be in its default, high-impedance input state for ports A, B and C. Thus random sampling of the TUBE (port A) by the BBC will return the instantaneous value of the 6809 data bus which could be anything. If the TUBE check is made before the Dragon is switched on however, the 6502 is forced to read all zeroes due to the pull-down action of RPK1.

BUYING A DRAGON

The cheapest way to obtain a Dragon computer is via the 'For Sale' columns of specialist computer magazines. Prices for the machines can vary greatly but it should be possible to buy a second hand Dragon 64 for under £100. Remember though, it is necessary to use a 64K computer in the second processor project. There are many 32K machines offered for sale but these will not be suitable unless fitted with an extra 32K of RAM (an article describing this conversion is scheduled for a future issue of *Electronics and Computing).*

The alternative is tò buy a new machine via Compusense, the UK distributor of Dragon products. The company can be reached on 01 882 0681. The price of a 64 is £195 inclusive of VAT.

The Intel 8255 is a popular peripheral, often used in straightforward control applications. It has two 8 bit ports, A and B, and two nibble wide ports, C lower and C upper. These can be configured on a group basis, as either input or output. This operation only involves writing to the appropriate registers within the 8255. This method of I/O is known as mode 0. Mode 1 offers the facility for operating port A in either input or output with handshaking functions built-in to three lines of port C. An extension of this principle is mode 2 whereby port A is suitable for bi-directional data transfer with five lines of port C used for the handshake protocols. This indeed is the mode used here, set up by the BBC writing $E2 to the 8255 control register.

A timing diagram for a typical data transfer is shown in **Figure 2.** Assume that the 6809 wishes to output a character to the

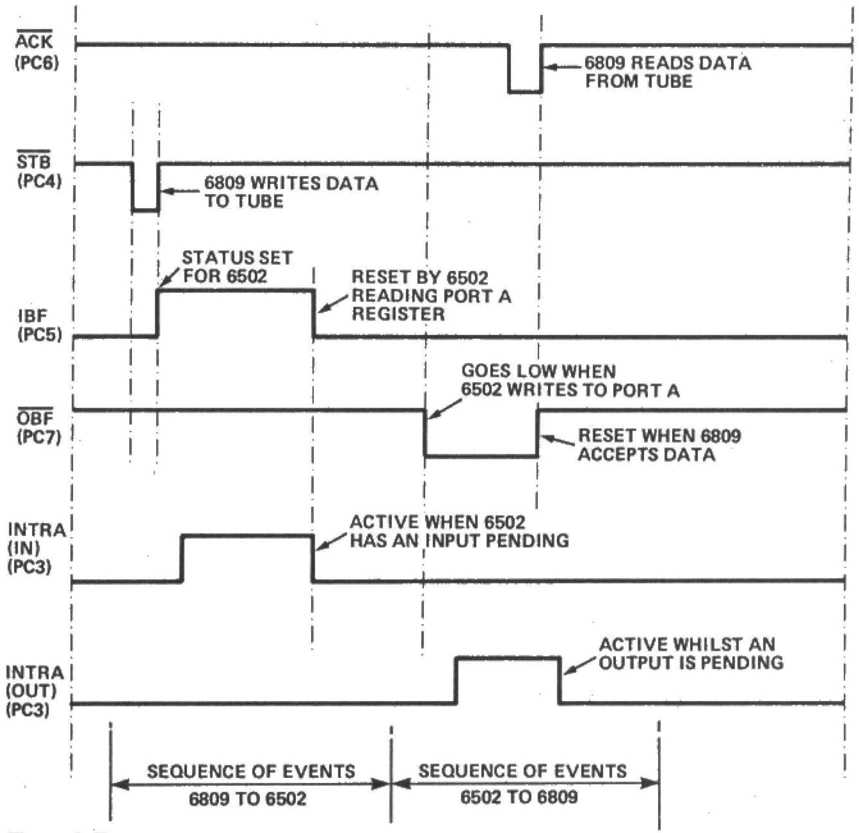

ACK (PC6)
6809 READS DATA FROM TUBE
STB (PC4)
6809 WRITES DATA TO TUBE
STATUS SET FOR 6502
RESET BY 6502 READING PORT A REGISTER
IBF (PC5)
GOES LOW WHEN 6502 WRITES TO PORT A
OBF (PC7)
RESET WHEN 6809 ACCEPTS DATA
INTRA (IN) (PC3)
ACTIVE WHEN 6502 HAS AN INPUT PENDING
ACTIVE WHILST AN OUTPUT IS PENDING
INTRA (OUT) (PC3)
SEQUENCE OF EVENTS 6809 TO 6502
SEQUENCE OF EVENTS 6502 TO 6809

Figure 2. TUBE data transfer.

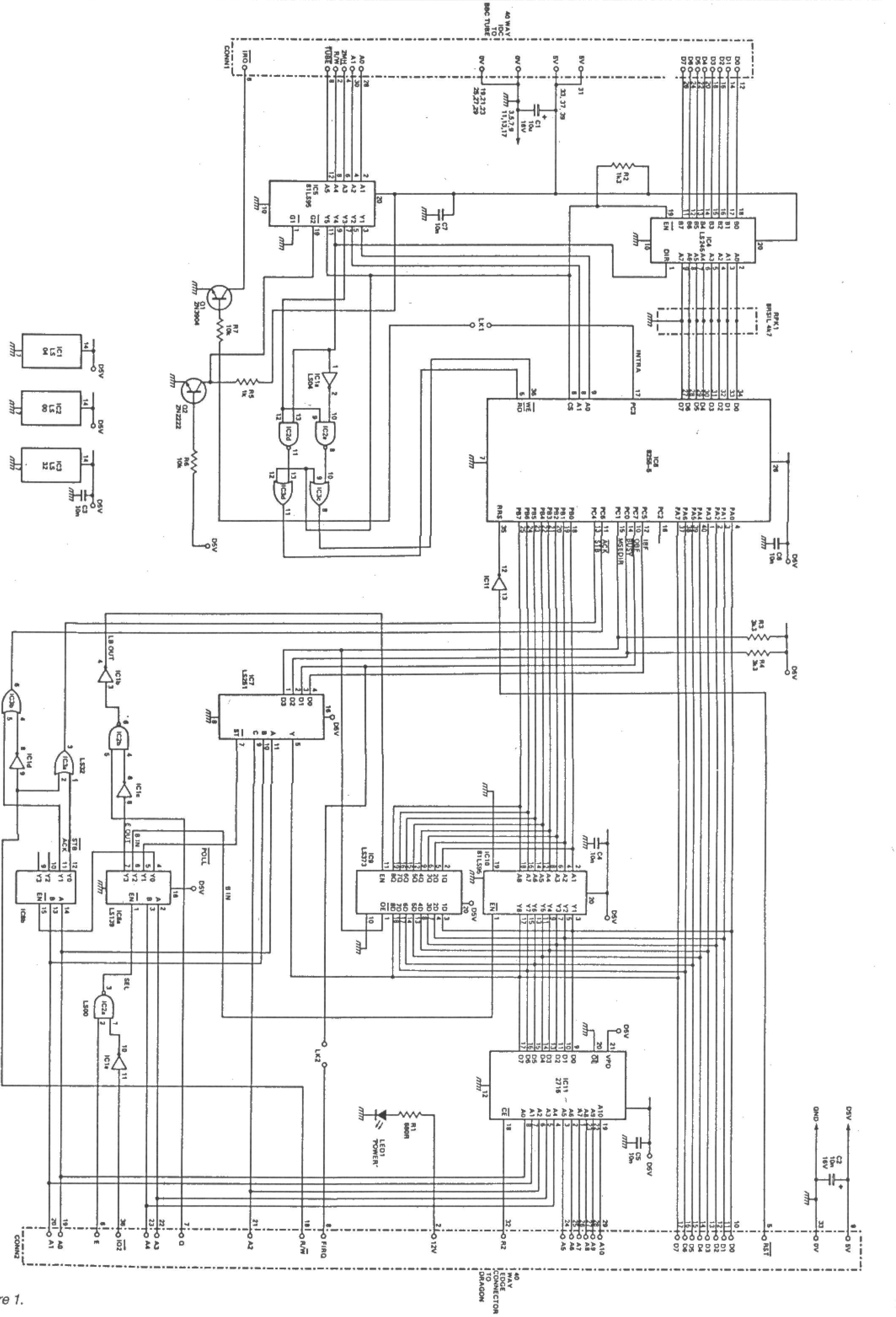

40 WAY IDC TO BBC TUBE
CONN1
IRQ
TUBE
R/W
2MHz
A0
A1
A2
0V
5V
D0
D1
D2
D3
D4
D5
D6
D7
C1 10u 16V
C7 10n
R2 1k3
IC5 81LS95
IC4 LS245
DIR
EN
RPK1 8RSIL 4k7
Q1 2N3904
Q2 2N2222
R5 1k
R6 10k
R7 10k
LK1
INTRA
IC1a LS04
IC2c
IC2d
IC3a
IC3c
IC8 8255-5
WR
RD
CS
PC3
RES
IC1 LS04
IC2 LS00
IC3 LS32
C3 10n
C6 10n
D5V
OBF
BUSY
IBF
ACK
STB
MSEDIR
R3 3k3
R4 3k3
IC7 LS251
LB OUT
IC1b
IC2b
IC1c
IC3b
IC1d
IC3a LS32
IC6b
IC6a LS139
POLL
B IN
E OUT
STB
ACK
SEL
IC2a LS00
IC1e
IC9 LS373
IC10 81LS95
C4 10n
C5 10n
IC11 2716
VPD
OE
CE
LK2
LED1 'POWER'
R1 680R
C2 10u 16V
GND
CONN2
A1
A0
E
IO2
A4
A3
Q
A2
R/W
FIRQ
12V
R2
A5
A6
A7
A8
A9
A10
RST
0V
5V
40 WAY EDGE CONNECTOR TO DRAGON

Figure 1.

BBC micro through the TUBE. It first writes the ASCII code to $FF40 which is decoded via IC8b, half an LS139, which produces a low pulse on the STB (PC4) input of the 8255 just as the data appears on the data bus at port A. This effectively latches the data into the 8255 and sets IBF (PC5) bit in the 8255 internal port C status register which signifies an input byte is ready. The 6502 detects this by polling $FEE2. An output, IBF, mirrors the status of this bit to the 6809, which tests its state by reading from $FF48. An LS251 data selector, IC7, passes IBF to D7 of the Dragon data bus. When the BBC acknowledges the character by reading the port A register, IBF automatically returns to the idle state (low) and the TUBE is available for further transfers. Overrun is prevented by the 6809 software which must always check IBF before releasing a character to the BBC.

For transfers in the opposite direction, the 6502 writes a data byte to the 8255 port A register ($FEE0) which is loaded into the 8255 output buffer and sets OBF (PC7) low to indicate to the 6809 that a character is pending. Again, the 6502 can monitor the current state of this output through the 8255 port C register. The Dragon reads the status of OBF from $FF49, recognising that a byte is waiting if it returns a positive value (below $80). It fetches the byte by reading from $FF41 which puts a low strobe pulse on ACK (PC6) to enable the 8255 output buffer which places the byte onto the second processor data bus via port A. INTRA (PC3) is an interrupt output from the 8255, linked to the 6502 IRQ line by Q1, an open-collector inverter. An interrupt could be made to occur on receipt of an incoming byte or when a character output sequence has completed.

With this scheme, port B and three lines of port C are redundant which suggests an auxiliary data transfer channel could be implemented. In fact a message port has been incorporated within the design, the only cost penalty being two LS chips. Transfer of data through port B can occur in either direction but the 6502 must always configure the mode of port B prior to the change of data direction. An output is assigned, MSDIR (PC1), which informs the 6809 whether port B is currently available for read ($FF58) or write ($FF50) and controls the OE, pin 1, of IC9, an LS373 latch used for data from the 6809 to the BBC. No conflict thus occurs with IC10, another 81LS95, which buffers port B data in the opposite direction until enabled via pin 1. A pull-up resistor, R3, ensures the LS373 is disabled whilst the 8255 is in its post-reset state. MSGDIR is tested by the 6809 through polling $FF4B. Another port C (PC0) is designated a 6502BUSY output, sampled on the second processor side by reading $FF4A – if positive then the 6502 is free. Note that it is the 6502 software which controls this signal. At the moment, MSGDIR and 6502BUSY are used to:-

(a) pass command completion semaphores from the 6502 to the 6809.

(b) synchronise the second processor to the host when it is engaged in processing a time consuming command such as reading from a disk.

the combinational logic, comprising gates 1a, 2c, 2d, 3a and 3d, convert the 6502 bus control signals into versions suitable for the 8255. It is important that a 5MHz part is used for the latter device in order to run at the 2MHz rate of the TUBE.

Link options are provided so that incoming data bytes could be serviced by interrupts on both CPUs. These are currently left open-circuit since interrupt driven transfers offer no substantial benefits and present some awkward problems on the 6502 side. A 2716, IC11, is used to hold the 6809 FLEX BOOT software which is copied across to RAM at $F800 in map type 1. The circuit around, and including IC8, generates the various signals needed to route any data and ensures certain addresses are read-only or write-only as appropriate. A 'power-on' LED is included for completeness.

NEXT MONTH
TUBE software and how to customise FLEX to operate across the TUBE and details of how to obtain the pcb and kit of parts.

Remedies for a sick disk drive

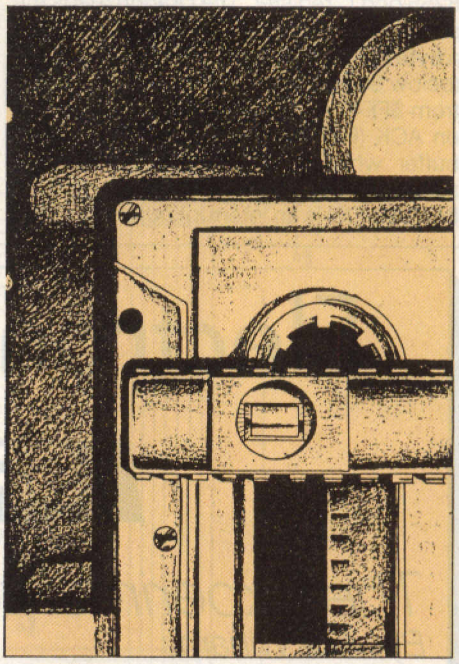
Too many disk errors on drive B? It may not mean a desparate call to the repair man if you follow Mike James' guide to simple disk drive operation and maintenance.

Most microcomputer users would consider floppy disk storage as the best compromise between cost and performance currently available. Cassette tape systems are cheap but too slow and/or unreliable. Hard disks are high capacity and fast but they are certainly not cheap and this leaves the floppy disk as the only contender for the accolade "Best Buy", being reasonably fast, reasonably cheap and reliable. The special status that the floppy disk occupies within the microcomputer world is illustrated by the common belief that a microcomputer that has only cassette storage is a computer waiting for disk drives to be fitted!

There are some applications for which floppy disks would be no more than a luxury but there is a lot of truth in the idea that serious computing only becomes possible when a pair of disk drives are included in the system. This is not to say that choosing and using floppy disk drives is trouble free: the acquisition of a pair of drives can be the start of a whole new world of trouble for the innocent micro owner! This article describes the basic mechanical principles that lie behind the operation of the floppy disk and the specific problems that arise in use. This hardware information should help you understand some of the faults that occur even in the best disk systems. Many floppy disk problems are however, connected with the other component of a computer system, the software, and next month's article describes how a microcomputer makes use of the disk hardware to store data and program files – in other words, it tackles the tricky problem of disk formats!

The story of the invention of the floppy disk is a strange one indeed. The first floppy disk was used to load the operating system into an IBM mainframe. In other words it was used to get the machine started (bootstrapped) and then it sat in the operator's console idle until the next time the machine was started up from cold. Fortunately for us smaller computer users, a number of people noticed the potential of the device as a low-cost fast data store and floppy disks started to be incorporated into minicomputers. As far as their use in microcomputers was concerned the original floppy disk was both too large at 8″ and too expensive and it was only with the development of the 5¼″ or minifloppy disk by Shugart that the floppy disk became a popular microcomputer storage device. In essence the 5¼″ device was nothing more than a scaled down version of the original 8″ device but because of its reduced performance (amount of storage and speed of access) it was easier and hence cheaper to make. Over the years though the performance of 5¼″ drives has improved while the cost has stayed the same or even reduced – this is mainly due to the large production volumes.

The principles that lie behind the operation of a floppy disk, be it 8″ or 5¼″, or even the new 3″ and 3½″ formats, are the same as those involved in any form of magnetic recording as described in detail in parts one and two of this series. That is, data is recorded as a pattern of magnetisation in a suitable material. This means that a floppy disk works in the same way as a tape recorder apart from the obvious mechanical differences – the magnetic material is in the form of a revolving disk rather than a tape and the read/write head is free to move. Data is recorded on a floppy disk in a number of concentric rings called tracks and the main advantage of using a disk is that the read/write head can move to any track and access the data without having to read any irrelevant data. (This should be compared to the problem of reading information stored in the middle of a cassette tape.)



Now while the detailed workings of a floppy drive are described in the next section it is worth contemplating the plausibility of the recording method! A floppy diskette is a disk of thin plastic coated with a magnetic material and then sealed in a flexible plastic envelope (see **Figure 1**). There are also various holes cut in the envelope to give access to the disk. In particular there is a central hole that allows a motor driven hub to engage with the thin plastic disk and spin it within the envelope. Get hold of a floppy disk and attempt to turn it by hand and it will feel very stiff indeed and spinning it at 360 rpm may seem a feat of great engineering (or foolhardiness!). If this was not enough in the way of rough treatment, while the disk of thin plastic is spinning a hard ceramic head is brought into contact with it so that data can be read and written. Now while the head only rests on the disk with a

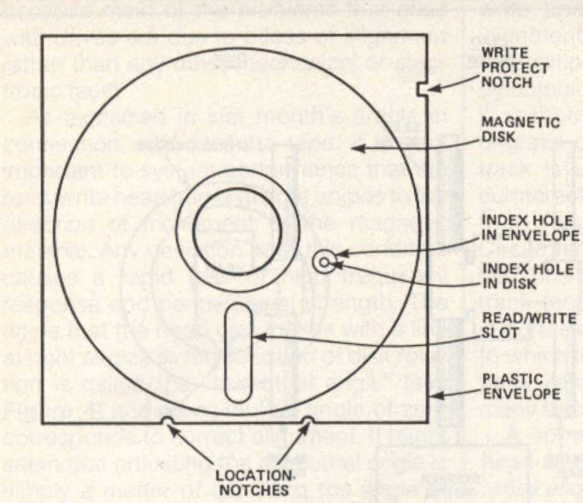


Figure 1. Anatomy of a floppy disk.

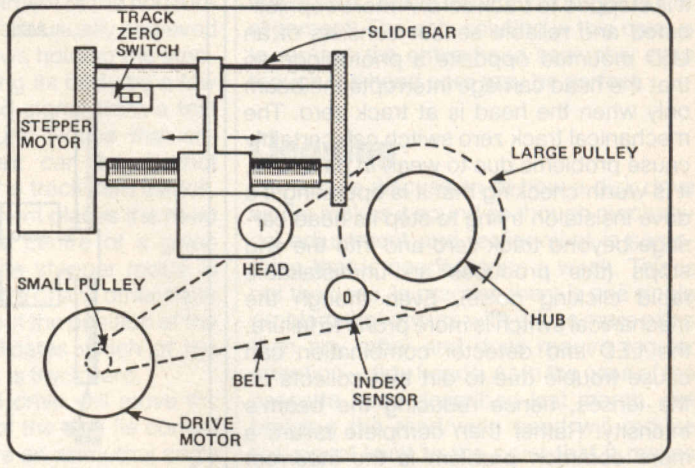


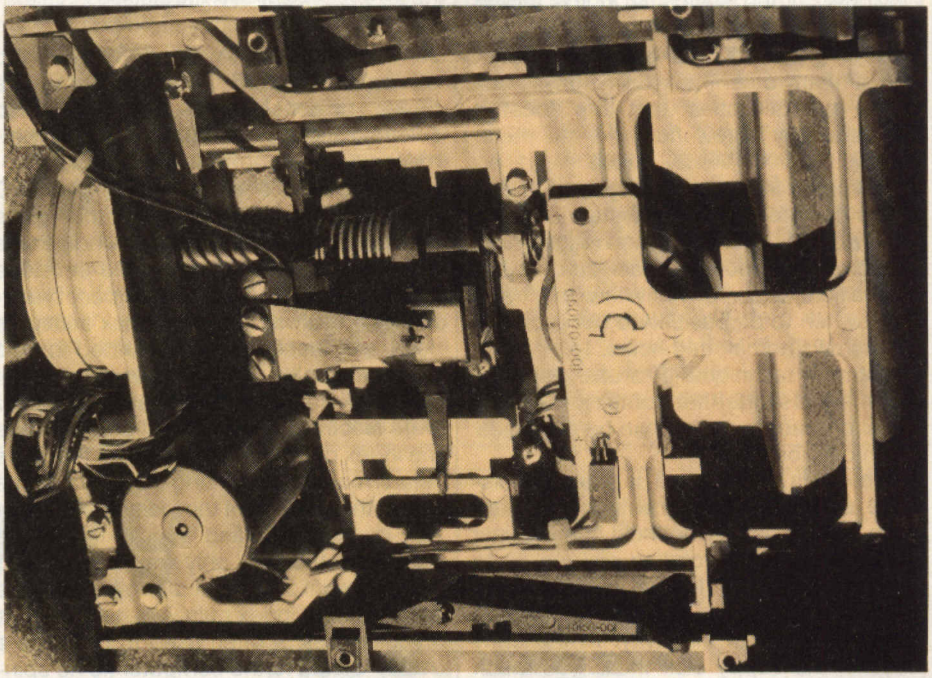
Figure 2. The main components of a disk drive.

weight of two grams (if your disk is correctly aligned) this is still a considerable shock to a thin plastic disk! And if both sides of the disk are being used it is pinched by a pair of hard ceramic heads, one on each side. After this description perhaps you will share the view that the floppy disk is a very unlikely device indeed and the fact that it works at all seems remarkable.

The hardware and its failings

All floppy disk drives share the same physical construction (see **Figure 2**). In this section the way the disk drive mechanics work and the typical faults to which they are prone are described. Difficulties with the read/write head are described later.

The hub and clamping mechanism engage with the large central hole in the diskette and rotate it at a precise speed – (300rpm for a 5¼" disk and 360rpm for an 8" disk). A common fault in many drives is the tendency to overclamp the diskette which results in damage to the central hole. Sometimes this can be corrected by adjustment of the disk loading mechanism but it is usually easier to learn to live with the problem and load disks with a gentle pressure. The hub is usually driven via a rubber belt from a motor mounted in one corner of the chassis but these days a low profile direct drive motor is sometimes used to save space. Belt slip or breakage is the main problem with indirect drive but this is a much rarer problem than you might expect. The speed of the motor is either locked to mains frequency or controlled by an electronic governor. If the speed is controlled electronically then there is usually some way of adjusting it and a strobe pattern can often be found on one of the drive pulleys to enable it to be tested and set. In practice consistent slow or fast running of a drive is rarely a problem because the data recovery circuits are generally speed tolerant. Erratic speed changes can be much more troublesome (showing up as an inability to read data back even though the read electronics checks out OK). The most common cause of speed variation is a worn or slack drive belt but other causes include faulty power supply; worn motor; bad diskettes; and faulty speed control electronics.



The lead screw head transport mechanism exposed.

The read/write head is usually mounted on a carrier that is free to slide along a track formed by one or two metal bars. The slider mechanism has a tendency to collect dirt of all types and occasionally a drive fault can be tracked down to excessive friction between the head carriage and the bars. The solution to this problem is to clean the bars using (as most manufacturers say) a lint-free cloth. In most case lubrication of the sliding mechanism is not only unnecessary it is positively harmful in that while it appears to solve the problem at first it causes dirt to collect at an even greater rate!

The head carriage is moved into position by a stepper motor. A stepper motor is simply a specially constructed motor that moves by a precise amount (degrees of rotation) for each pulse applied to it (see *E&CM* July 1984, "Motors Explained"). By applying a given number of pulses to the stepper motor the head can be moved accurately to any position. There are three main methods used to connect the stepper motor to the head carriage: a metal band, a lead screw and a disk with a spiral track, each of which can be seen in **Figure 3.** All three methods work well in practice although the lead screw is often said to be the most accurate and trouble free. The spiral disk method of positioning does have one common fault that it is worth knowing about. It is possible for the ball bearing and spring assembly to be forced out of the slot by vibration or by attempting to move the head too far in any direction. Normally the ball bearing will automatically re-engage with the slot as the spiral disk rotates but sometimes the head can move out of range of the spiral and then the only solution is to open the case and move the head back into position by hand.

To be able to move the head to the same position more than once it is necessary to have a known and fixed starting position. This is usually the position of track zero (see later), the outermost track on the disk. All drives incorporate some kind of sensor that will let the controlling electronics know when the head is positioned over track zero. The simplest and cheapest form of track zero sensor is a mechanical switch

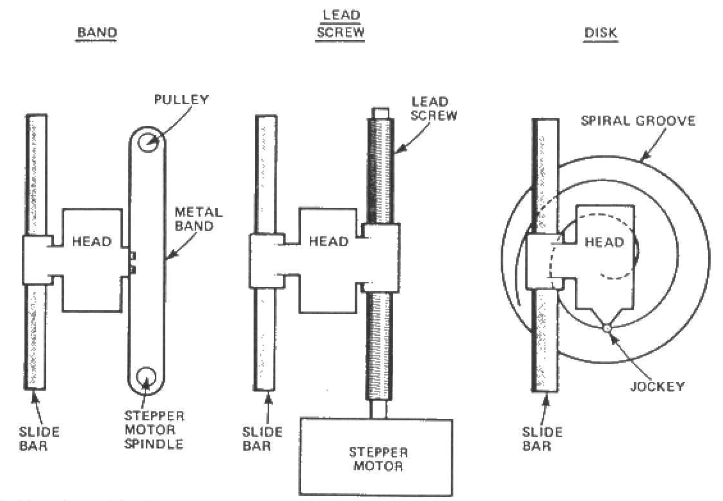


Figure 3. Head positioning methods.

that is operated by the head carriage when it is stepped to track zero. A more sophisticated and reliable sensor consists of an LED mounted opposite a photodiode so that the head carriage interrupts the beam only when the head is at track zero. The mechanical track zero switch can certainly cause problems due to wear. In particular, it is worth checking that it is operating if a drive insists on trying to step its head carriage beyond track zero and into the end stops (this produces an unmistakable rapid clicking noise). Even though the mechanical switch is more prone to failure, the LED and detector combination can cause trouble due to dirt that collects on the lenses, hence reducing the beam's intensity. Rather than complete failure a more common problem is the incorrect positioning of the track zero switch. Obviously if the switch is in the wrong place the head will not be positioned over track zero and will be unable to read data from it or any other track! Some drives use track zero switches that are located accurately by screw holes in the metal chassis and in theory these need no adjustment. In practice slight misalignments can creep into the system and it is often worth slackening off and retightening the screws that hold the track zero switch in position. Other drives allow the track zero switch to be positioned using one or two screws and slotted holes. The proper adjustment technique is to use a special alignment diskette that contains a steady signal on track zero that can be displayed on an oscilloscope.

When it is not required, ie when the diskette is not being read or written, the read/write head is usually not held tight against the surface of the rotating disk so as to reduce media wear. Indeed in many drives there is the facility to switch the motor off during periods of inactivity. In the early days of the floppy disk there was much debate about whether it was better to leave diskettes spinning all the time or start and stop them as required. These days continuous operation has become the norm, possibly because the time wasted waiting for a disk motor to come up to speed (anything up to 2 seconds) isn't worth the savings in diskette wear. However head loading and unloading is most definitely worth the effort and most disk drives have a head load solenoid. In a single sided drive, ie a drive with a single read/write head, the head load solenoid controls the position of a felt pad on the opposite side of the diskette.

The head load solenoid and its associated tension springs can cause trouble in a variety of ways. A completely burned out solenoid makes it impossible to load the head; an overtight spring can cause excessive wear; an undertight spring will allow the head to leave the disk's surface and reduce the output of the read head and so on. Single sided drives occasionally have to have the felt pad replaced due to wear or dirt. Double sided drives are particularly sensitive to head loading problems in that an over strong spring can cause the pair of heads to hit the surface of the disk with enough force to damage it. It is worth mentioning that some drives dispense with the head load solenoid altogether and keep the head permanently in contact with the diskette.

The index hole sensor is simply an LED and photodiode combination that gives out a pulse every time the index hole in the diskette passes the hole in the envelope (see **Figure 1**). There are a number of variations on the way that the index sensor is used and these will be described in next month's article. The most important use of the index sensor is to signal to the read/write electronics the start of a track. Thus the index pulse is a timing signal that says "start reading or writing now". In some drives the position of the index hole sensor can be adjusted and should this prove necessary then the only way to do it is to use a special alignment disk (see later). As the stream of index pulses generated by a spinning disk is often used to generate a 'ready' signal to the computer (ie no spinning disk-drive not ready) an index pulse sensor failure usually shown up by a "DRIVE NOT READY" type error message even though there is a diskette present in the drive. For the drive to work the diskette has to be inserted so that the index hole in the envelope corresponds to the position of the index hole sensor. As the index hole sensor is offset to the right this means that you cannot put a diskette in any way up – ie you cannot turn a diskette over and use both sides. Some drives overcome this problem by having two index hole sensors offset equally to the right and the left allowing the diskette to be turned over. Another method of using both sides of a diskette is to punch an extra index hole in the envelope so that when the diskette is turned over the index sensor can still detect the passing of the index hole (see *E&CM* December 1984 "Floppy Turnover"). It is also worth mentioning that the position of the index hole in an 8" disk is not entirely standard.

The final sensor found on all drives is the write protect sensor. This detects the presence or absence of the write protect notch in the corner of a diskette. Nearly all write protect sensors these days are LED/photodiode combinations but some early drives used mechanical switches and these were less reliable and prone to jamming. It is worth noting that a 5¼" disk is write protected if the notch is covered but an 8" disk is write protected if the notch is uncovered!

Headaches

The entire disk drive mechanism is concerned with putting the read/write head in the correct place over a rotating disk and as might be expected the most common disk drive fault comes down to the head being not quite where it should be. In other words, as in the case of cassette tape (see last month's article), head alignment is all important. Of course as a disk read/write head is free to move, alignment is correspondingly more complicated. Even so it is worth knowing about disk head alignment

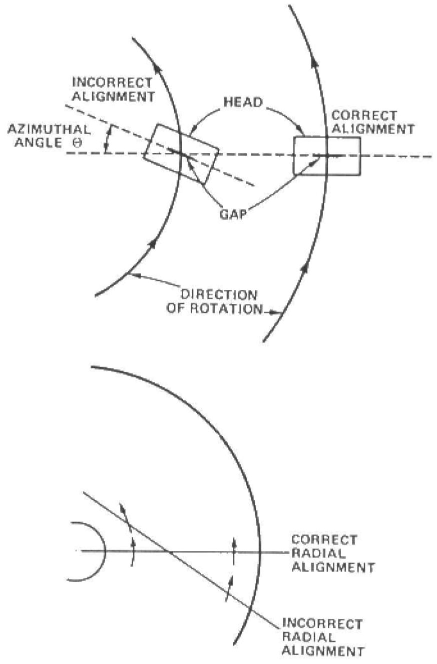


Figure 4 (top). Azimuthal alignment.
Figure 5. Radial alignment. The azimuthal angle varies along a line that is not a radius.

because most of the problems that arise with drives are due to a loss of alignment rather than any other mechanical or electronic fault!

As explained in last month's article in connection with cassette tape, it is very important to system performance that the read/write head gap is at right angles to the direction of movement of the magnetic material. Any deviation from this condition causes a rapid loss of high frequency response and hence signal strength. The angle that the head gap makes with a line at right angles to the direction of disk rotation is called the "azimuthal angle" (see **Figure 4**) and an azimuthal angle of zero corresponds to correct alignment. It might seem that adjusting the azimuthal angle is simply a matter of adjusting the angle of the head within the head carriage but this is in fact not sufficient to ensure that the azimuthal angle is the same for all positions of the head. For the azimuthal angle to be constant it is essential for the head to move along a line that passes through the centre of rotation of the disk, that is along a radius of the disk – see **Figure 5.** This is called "radial alignment" and is the first requirement for a disk drive to be able to read and write all the tracks of a diskette with the same level of reliability. If you have a drive that tends to give read/write errors always in the same group of tracks then suspect the radial alignment of the head. As well as azimuthal angle the read/write head must be positioned as closely as possible over the centre of any track it is trying to read or write and this is called "circumferential alignment". The adjustment of the horizontal position of the head is usually achieved by unlocking the screws holding the stepping motor and rotating its body by a few degrees until the read signal from a test track is at a maximum. Notice that circumferential alignment can be distinct from the position of the track zero switch. Circumferential alignment places the head into position over the centre of a given track (and hence if the stepper motor is accurate over the centre of any other track to which it is moved) but the position of the track zero switch indicates which of the many tracks on a disk is track zero.

A correctly aligned drive will move the head along a radius of the disk (ie correct radial alignment), have an azimuthal angle of zero (ie correct azimuthal alignment), will position the head over the centre of any track (ie correct circumferential alignment) and the track zero switch will correctly identify the position of track zero. When aligning a drive it is usual to adjust the circumferential alignment, followed by the radial alignment, then set the azimuthal angle and finally the track zero switch. In practice each adjustment usually throws the previous adjustments out of position and it is usual to go through the alignment cycle a few times until no changes are necessary – the result being a perfectly aligned drive. Examination of any given drive will reveal that not all of the above alignments are adjustable, many will be factory set. Indeed there are some disks which have no head adjustments at all but this doesn't mean that they don't go out of alignment! The only solution in this case is to replace the entire head assembly even though the head core may be perfect.

Cleaning

The above discussion of how a disk drive works makes it sound as though mechanical adjustment and realignment is something that is needed once a week. This is not the case. In practice there is one single problem that occurs with drives more often than any other and does require regular attention – dirty heads. As in the case of the cassette head described last month, dirt bridging the read/write head will reduce the signal level to the point that it makes the drive look completely dead. The solution is simply to clean the heads using one of the many head cleaning diskettes on the market. Most read/write errors disappear and drives come back to life after a few minutes with a head cleaning diskette. Sometimes though, the dirt is so stubborn that it clings to the head no matter how long the cleaner is left in place. The only solution in these cases is to take the drive to pieces and clean the head directly using alcohol and a felt pad of the type used to clean video tape heads. If after this treatment a drive still fails then there is no choice but to look at the details of its alignment and its electronics.

Next Month – *How a micro uses the disk hardware.*

NEXT MONTH

QL CAD

Sophisticated software that turns the QL computer into a versatile aid to the production of circuit diagrams.

BBC time code generator

Professional audio and video tape machines have an in-built time code system that allows sections of material to be accurately timed. Paul Beverley shows how the BBC micro can be used to emulate the facilities of these costly systems.

The ST saga continues

We'll be twisting the arms of a few people during the next month in an effort to get our hands on an ST computer complete with 'working' versions of the software to be bundled with the machine. A full report on our findings in next month's issue.

Plus

Making music on the Amstrad, the Memotech computer that thinks it's a Spectrum, Comms News and all our regular features.

SEPTEMBER ISSUE ON SALE AUGUST 13th

Place a regular order at your newsagent

IN BRITAIN'S BEST SELLING COMPUTER PROJECTS MAGAZINE

Please Note: Content is subject to late revision.

BETTER BASIC WITH WORDWISE +

BBC owners jealous of SuperBasic's structured programming facilities can hop on the bandwagon if they have Wordwise Plus. Paul Beverley explains how.

Those virtuous(?) people among you who believe in structured programming, will find using Wordwise Plus for writing and editing programs a great help. By writing programs without using GOTOs, and putting AUTO at the top of the program file, a program without line numbers can be SAVEd with option 1. By then going into BASIC by using *B. <return> and bringing the file from disk or tape by using the *EXEC W <return> (or just *E. W), the program text comes in and is given line numbers automatically. When it has all come in it is possible to list the BASIC program by pressing escape.

It is perfectly possible to write BASIC programs without using line numbers. This is not really the time or place to explain the techniques, but there are a number of books available about structured programming.

BBC BASIC has enough features to enable structuring to be implemented, but one problem when trying to work entirely without line numbering is how to do error trapping routines without using ON ERROR GOTO. In fact it is easy to get round the problem – you can use:

```
ON ERROR PROCerror: END
```

Then somewhere amongst the procedures in a program a DEF PROCerror section is added, an example is shown in **Listing 1.** If the program uses graphics, it will be necessary to do a mode change to make the error report readable. As it is not possible to change mode within a procedure, the following should be used:

```
ON ERROR MODE7:PROCerror:END
```

One major advantage of writing programs in a structured way, entirely without using line numbers, is that it is possible to edit them in Wordwise and reorganise the program to make it more readable and logical by moving the different parts of the program around. If the program is properly structured, this can be done freely because it is immaterial where the definition of any given procedure appears.

Another advantage of using Wordwise for editing is that global variable name changes can be carried out, providing care is taken not to end up with two variables with the same name. In any case, when editing the program, it is possible to go to the top of the text and use the search and replace facility to check whether a particular variable already exists in the program.

An existing program can be entered into Wordwise ready for editing by following the steps shown in **Listing 2.**

LISTO0 makes sure that there are no added spaces in the program listing, and any existing file called "W" is deleted to avoid getting "Can't extend" if the new file is bigger than the old one. This only applies to disk users, tape users can omit this line. By calling the file "W" it is easy to decide which files are temporary and thus no longer required.

If this facility is to be used regularly it's probably worth programming a function key with the following sequence:

```
*KEY 1 L.O0¦M*DEL.W¦M*SP.W¦ML.
¦M*SP.¦M*W.¦M2W¦M
```

A problem occurs though if the BASIC program is a very short one. In this case, if Wordwise has just been used it is possible that the existing file in memory is still intact. It will then ask "Are you sure? (Y/N)" and the "W" in the function key definition will be counted as "No" so that it will revert to the menu. In this case press <2> again, and then <Y> and <W> <return>.

Once the program text is in Wordwise it can be edited as it stands, with the line numbers, or these can be stripped out using a function key, thus:

```
*KEY 3¦!$¦M¦A¦M¦A¦A¦A¦A¦A
```

This moves the cursor to the carriage return at the end of the current program line (¦!$¦M), deletes the carriage return (¦A) and then re-inserts it (¦M). This has the effect of bringing the cursor down off the

LISTING 2.

```
LISTO0 <ret>
*DEL. W <ret>
*SP. W <ret>
LIST <ret>
*SP. <ret>
*W. <ret>
<2>
W<ret>
```

LISTING 1

```
DEF PROCerror
VDU 3        (Switch off the printer, if you're using one.)
REPORT
PRINT " at line ";ERL
ENDPROC
```

end of one line to the beginning of the next. The finally it does ¦A five times to take out the line number and the spaces before it. Holding down <shift-ctrl-f3> and the auto-repeat then strips out all the line numbers from top to bottom. Holding the keys down for too long will mean the end of the program will be reached and Wordwise will 'buzz' indicating an attempt to delete off the end of the file, and extra carriage returns are added, but these can then be deleted. In fact it would not matter if they were left in because when the file is taken into BASIC using AUTO, BASIC would simply treat them as null lines.

One problem here is that as it loads in the spooled files Wordwise finds a linefeed as well as a carriage return. Since it does not recognise the linefeeds it replaces each of them with a pad character, <¦>. These could be removed by a global search and replace, but if the program contains any function key definitions that use this character for control codes, they will be removed as well, One alternative is to extend the function key definition as follows:

```
*KEY2¦!$¦M¦!,¦A¦A¦M¦A¦A¦A¦A¦A
```

This goes to a carriage return, moves back one space to be under the pad character and then deletes it with <ctrl-A> before carrying on as the previous key definition.

It is possible however do the whole thing, including stripping out the line numbers, using a segment program as shown in **Listing 3.**

If this is loaded into segment 0, say, then when <shift-f0> is pressed nothing much appears to happen for a few seconds. Finally a bleep will be heard as the program tries to delete some non-existent characters at the end of the file. This signals the end of the segment program. To see it all happening, simply put a DISPLAY instruction somewhere in the loop, but note that this increases the run time by about 50%.

The programmed keys that complement the segment program are shown in **Listing 4.**

Having loaded the program in BASIC, press <f1>. This spools the program with the title W. Function key 2 then switches to Wordwise and loads in the segment program called "S.BASTRIP" which must be on the current disk. An alternative would be to keep S.BASTRIP on a disk in a second drive (if available) and call the file by using ":1.S.BASTRIP". The ¦!¦@ generates <shift-f0> and therefore runs the segment program. The program bleeps when it has finished and switches back to the normal menu, so the program can be edited by pressing <escape>. Then finally pressing <shift-ctrl-f3> saves the file and *EXEC's it back into BASIC.

LISTING 3.

```
SELECT text
LOAD "W"
CURSOR top
DELETE AT 3         deletes ">L."
TYPE "AUTO"         enter "AUTO" ready for BASIC
REPEAT              Repeat
  FKEY 4,CHR$13     move to carriage return
  DELETE LEFT 1     delete the pad character
  DELETE AT 1       delete the carriage return
  TYPE CHR$13       put it back to bring cursor
                    to new line
  DELETE AT 5       delete spaces + line number
UNTIL EOT           until end of text
```

LISTING 4.

```
*KEY 1 L.O0¦M*DEL.W¦M*SP.W¦ML.¦M*SP.¦M
*KEY 2 *W.¦M2W¦M92S.BASTRIP¦M¦!¦@
*KEY 3 1W¦MY*B.¦M*E.W¦M
```

If the lines of a program are longer than 40 characters, Wordwise will split them up into what it considers are words, ie splits the line where spaces occur. This can make the program a little difficult to read but by pressing <ctrl-F>, Wordwise changes the screen formatting and makes the program easier to read.

Listing PROCs and FNs

When writing a long program in BASIC, it is often useful to have a list of procedures and functions, particularly when it comes to documenting the program. Here is a recipe:

a: Spool the program into a text file using LISTO0 and then switch to Wordwise and load it in.

b) Load the following program into segment (0) and then press <shift-f0>.

Note – the line numbering of **Listing 5** is purely for discussion of the program. You should not type them in.

Lines 7 to 10 pick up occasions where the programmer has used a space between DEF and PROC or DEF and FN, and changes them so that all functions and procedures are in the form DEFPROC and DEFFN.

Formatting assembler listings

To produce formatted listings of assembly code programs simply use the backslash character as usual for your comments, get the program into Wordwise by spooling it as shown above (though perhaps with LISTO7 to show the program structure) and then use a function key to add tab characters in front of each backslash:

LISTING 5.

```
 1 REM Function + Procedure
 2 REM Listing Program
 3 REM (C) 1985
 4 REM Norwich Computer Services
 5
 6 SELECT TEXT
 7 CURSOR TOP
 8 REPEAT
 9   REPLACE "DEF ","DEF"
10 UNTIL EOT
11
12 CURSOR TOP
13 FKEY 3
14
15 REPEAT
16   REPLACE "DEF",""     Delete DEF
17   CURSOR AT 0          Move to left
18   FKEY 3               Put in a marker
19   FKEY 7               Delete marked text
20   CURSOR AT 0          Move to left
21   FKEY 3               Marker
22   CURSOR AT 6          Move past line number
23   FKEY 3               Marker
24   REPLACE "¦","¦T"     Replace ¦ with tab
25   CURSOR RIGHT         Move past tab
26   FKEY 8               Move line number onto end of line
27   DISPLAY              Optional display
28   CURSOR DOWN          Move down
29   CURSOR AT 0          Move left
30   FKEY 3               Marker
31 UNTIL EOT              Until end of text
32
33 DELETE LEFT            Remove unused marker
```

Thus:

```
*KEY 7¦!$\¦I
```

This moves the cursor to the next backslash and adds a tab character (I). An appropriate DT command can then be added to get all the comments into just the right horizontal position.

It is also possible to tabulate the mnemonic codes themselves and leave the labels with a negative indent so that they show clearly. Assuming that the program is renumbered in 10's, use a global change and replace <0><sp><sp><sp><.> by <0><sp><.>. Then if that is not a big enough indent for your liking, do a global change of <0><sp><sp><sp> into <0><sp><sp><sp><sp><sp><sp> which should be enough.

This could also be done with the segment program shown in **Listing 6.** (Again, do not type in the line numbers.)

LISTING 6.

```
 1 SELECT TEXT
 2 LOAD TEXT "W"
 3
 4 CURSOR TOP
 5 REPEAT
 6   FKEY 4,CHR$13
 7   DELETE LEFT 1
 8 UNTIL EOT
 9
10 CURSOR TOP
11 REPEAT
12   REPLACE "\","¦T\"
13   CURSOR RIGHT 2
14 UNTIL EOT
15
16 CURSOR TOP
17 REPEAT
18   REPLACE "Ø   .","Ø ."
19   CURSOR RIGHT 2
20 UNTIL EOT
21
22 CURSOR TOP
23 REPEAT
24   REPLACE "Ø   ","Ø¦T"
25   CURSOR RIGHT 2
26 UNTIL EOT
27
28 CURSOR TOP
29 TYPE "¦GDT11,35¦R"
```

Lines 4-8 strip out the pad characters that Wordwise inserted as it loaded the spooled file. Lines 10-14 put a tab character in front of each backslash in order to tabulate the comments. Lines 16-20 pull the labels back towards the line numbers. Lines 22-26 put a tab in front of each mnemonic code so that they can be tabulated to taste. The last two lines put in the tab stops, the values of which can of course be altered according to taste.

This article is an extract from one of the Chapters of Paul Beverley's 'Using Wordwise Plus', available through Norwich Computer Services – telephone 0603 621157.

How to modify an OS9 serial printer driver

Atari has confirmed that the 520ST will retail for £749, but the promised machines, due in the shops in limited quantities by late June, are yet to appear.

There are strong rumours that Atari is suffering from Sinclair's disease, otherwise known as cash flow problems. Evidence to back up the rumours was provided first at the Chicago CES, where Atari took only a tiny stand to show off the ST, and then by the closure of parts of Atari's Hong Kong facilities to ease the financial demands on the company.

There are already some 100 STs in the UK, but at the time of writing these are available only as development packages for software houses: none are available in the shops. It is reported that there are some problems with the implementation of BASIC on the machine and Atari have, quite rightly, decided that it would be very difficult to attempt to market a computer without some form of BASIC interpreter. If and when the machine becomes available the 520ST will be sold with the Gem OS, Basic, Logo, Gem Write and Gem Paint all on disk. ROM based software will be available by September at the earliest. The 250K version will (says Atari), be ready for the Christmas market and priced at between £300 and £400.

Included in the 520ST's £749 price will be a 12" hi res mono monitor (the ST mark 1 has no RF modulator – as reported by *E&CM* last month), a mouse, one 0.5Mbyte 3.5" disk drive, and of course the software. This is an excellent deal but early buyers should remember that it will almost certainly be impossible to upgrade disk based GEM to the ROM based version. This could have important implications both in the amount of RAM available for applications programs in early versions of the ST and in a degredation in the speed of operation.

Still no Atari 520STs in the shops

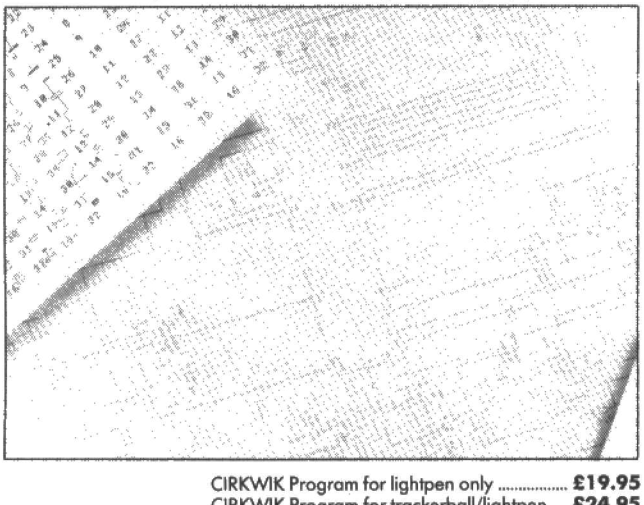
The OS9 operating system supplied by Dragon Data provides a serial printer driver routine called ACIA51. Anyone who has tried to use this program in conjunction with a printer will know that, unfortunately, it is none too reliable in operation. The problem is that when trying to send data to a slow device such as a printer via the RS232 serial port, not every character reaches the printer. While – in theory – the printer has control of the transmission process via its DTR signal, the way in which ACIA51 is implemented blows theory to the winds.

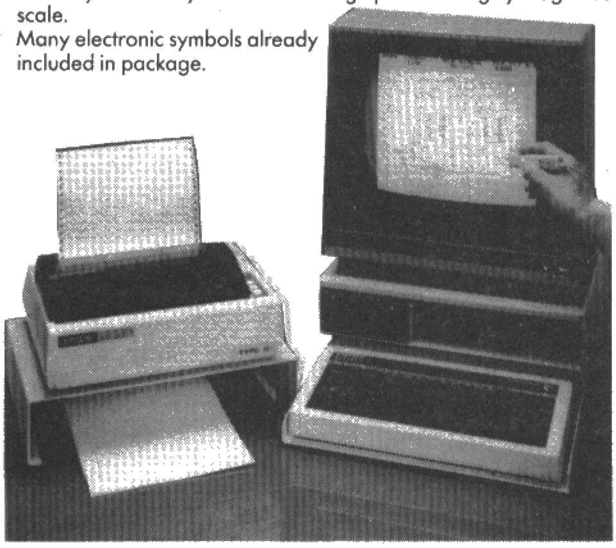
The RS232 driver IC in the Dragon is a SY6551, under normal conditions, whenever it requires the next character to send to the printer, it sends an interrupt to the 6809 CPU and sets a character in its control register to indicate that the transmit register needs refilling. The CPU responds by identifying the cause of the interrupt and writing the next character to the transmit register. The problem with Dragon Data's implementation of the driver software is that the DTR/CTS line goes from a 'ready for data' to a 'not ready for data' state in the interval between the 6551 raising the interrupt flag and the CPU responding to it. The character written to the transmit register is consequently lost.

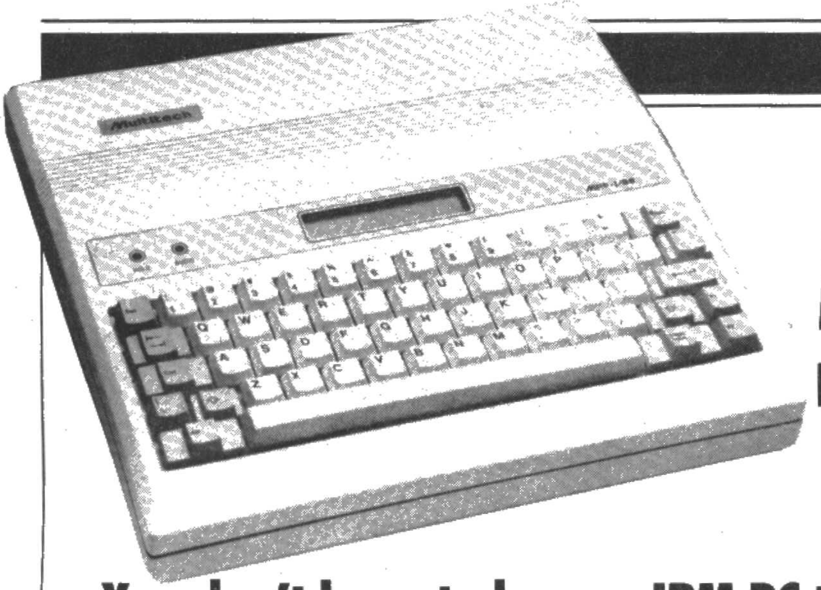
The new device driver gets round the problem by double-checking the transmit register before sending a character. With this alteration the ACIA51 driver gives reliable operation.

The author will supply any *E&CM* reader with the new driver free of charge. Those wishing for a copy of the modified serial printer driver should send a formatted OS9 disk to the address below together with a stamped addressed envelope.

Serial Driver, 22 Greenacres,
South Cornelly, Bridgend, MID GLAMORGAN, CF33 4PT.







MPF-1/88

You don't have to buy an IBM PC to learn 8088 machine code programming. Sue Gee reviews this low-cost introduction to 16-bit hardware.

In this day of low-cost microcomputers such as the Spectrum, Electron and Amstrad you might wonder why anyone would be interested in buying a system that, on paper at least, seems very limited for around £300. The reason is that if you are interested in learning about how microprocessors work the average microcomputer has just too many features to enable you to get anywhere near the hardware and the machine code that lies inside it. Hence the need for specially designed "micro-trainers" that, rather than keeping the user away from the hardware and machine code, actually demand an involvement.

The Microprofessor series of micro and digital trainers from Flight Electronics is a good example of such equipment and the earlier Z80 based system has already been reviewed in *E&CM* (November 1983). Even in the world of trainers however, things move on and the latest offering is the MPF-1/88 using an 8088 microprocessor, one of the most popular, although not state-of-the-art, 16-bit microprocessors. (The status of the 8088 as a 16-bit processor is arguable as it still uses an eight-bit data bus and hence is better thought of as an 8/16-bit processor.) Perhaps the most important thing about the 8088 is that it is the microprocessor to be found inside the IBM PC and most other IBM compatible machines. Because of its use of a 16-bit micro the MPF-1/88 will be of interest to a wide audience. For those that know about traditional eight-bit micros it provides an opportunity to experiment with the more up-to-date 16-bit devices.

'The expansion connector is compatible with the IBM PC's I/O expansion slot'

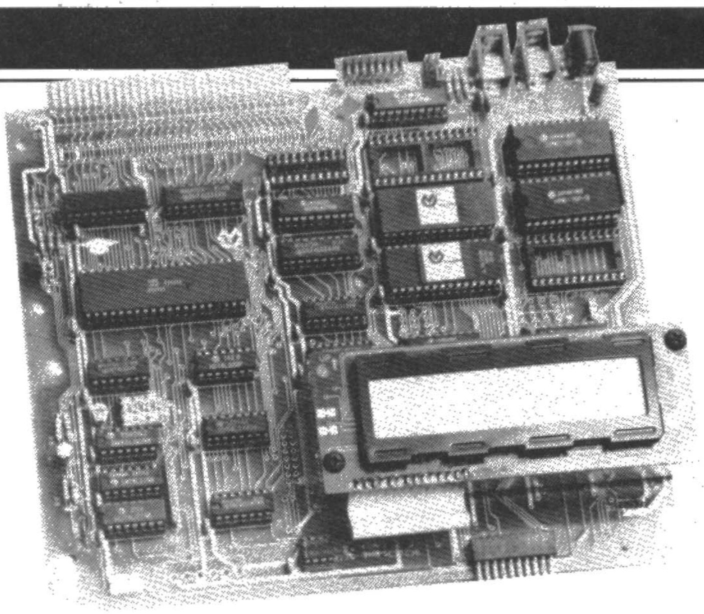
A brief technical run down on the MPF-1/88 indicates that, while it might use a 16-bit microprocessor, its standard ROM and RAM capacities are in the tradition of the eight-bit trainers – 8088 4.77MHz CPU, eight-bit data bus, 4K RAM, 16K ROM, two line 20 character LCD, full 59 key keyboard, printer port, cassette port and a 62 pin IBM compatible external bus connector. If you are using the MPF 1/88 as a trainer then 4K of RAM is more than enough for your programs and the 16K ROM contains all of the system routines and development software likely to be needed. The RAM can be expanded to 24K by plugging in extra chips and similarly the ROM can be expanded to 48K. More RAM than this can be added (the 8088 can address 1M byte) via the expansion connector.

The MPF-1/88 certainly does not look like a traditional style trainer, for one thing it is in a case that would do justice to any machine. The back half of the case pulls off with remarkable ease to give access to the RAM/ROM and other devices. The keyboard is a standard full function QWERTY layout and the display is a two line 20 character LCD device which is a window onto a 24 line by 20 character logical display screen. In practice the display is adequate for working with programs of 20 to 30 lines but the optional thermal printer is essential for anything bigger (or for students who lack confidence). A cassette deck is used for program storage in the usual way – it proved reliable in operation. Apart from the parallel printer port and a buzzer, the only other I/O connector on the machine is the expansion port which is really the key to doing anything adventurous!

The expansion connector is compatible with the IBM PC's I/O expansion slot connectors. It is compatible but not identical in that it has two extra connections and a number of minor differences – on the IBM PC pin A1 is I/O channel check, pin B5 is –5V and pin B8 is unused but on the MPF-1/88 they are hold, interrupt acknowledge and interrupt request respectively. However these differences are fairly minor and it should be possible to use IBM interface cards with only a little (if any) hardware trouble, but software to make use of them is another matter! Flight Electronics say that they have or will soon have a number of IBM style cards to use with the MPF-1/88 including a colour high res/text card, a 48 line I/O plus timer card, a PIO/CTC card and a dual RS232C interface card. No doubt enterprising users will soon add to this list by modifying and using standard IBM cards. Finally it is worth saying that while the expansion connector is nothing more than a section of the main PCB sticking out of the back of the case, just inside is a PCB layout that looks just right for taking an IBM style PCB socket.

The only software that comes with the MPF-1/88 is a system monitor in ROM. This contains all of the usual facilities that would be expected in a low level machine monitor – memory change, change/examine register values, run a program, single step a program etc – plus the two extra features – a single pass (immediate mode) assembler and disassembler – that are required for its role as a trainer. The assembler is as standard as an immediate line assembler can be in that is supports all the 8088 (and 8086) mnemonics but of course you cannot use address labels etc. Flight Electronics promise a full two-pass assembler, BASIC and Forth. None of these promises sounds far fetched as there are plenty of suitable software packages for the 8088 and all that is needed is a little customisation and testing. In its training capacity it would seem that the most needed piece of software is the two-pass assembler which will allow a student to move on to using more realistic development software after coming to terms with the underlying hardware using the line assembler. All of the important subroutines within the monitor ROM are documented in the User's Manual and this makes tasks such as printing characters, reading the keyboard etc relatively easy.

The system documentation is very good indeed and consists of three separate manuals. The Users Manual is an introduction and tutorial on setting up and using the



The lid slips off easily to give a view of the MPF-1/88 pcb. The bus connector is at the top of the picture.

MPF-1/88. It describes how to run a program in simple terms but it doesn't make any attempt, quite rightly, to reach 8088 assembly language. For this it will be necessary to acquire one of the books recommended in the Reference Manual. There is plenty of information within the User Manual however, including a description of the monitor commands, memory and I/O maps, list of 8088 mnemonics etc. The real technical information is found in the MPF-1/88 Reference Manual which includes details of interrupt handling and a complete circuit diagram. The final manual is simply a complete and very welcome listing of the entire monitor program. This should make it possible to use many of the undocumented subroutines in the monitor and resolve any ambiguities about the way a subroutine works. It also provides an excellent and very long example of how to write 8088 assembly language! Flight Electronics also have a student work book, not available for this review, as an optional extra. However it should be easy enough for a lecturer to put together a few introductory notes from the three manuals, mainly the User's Manual, that will enable a student to work with the MPF-1/88.

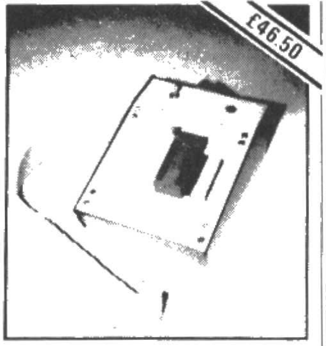
The MPF-1/88 is ideal for use in small groups to teach fundamental microprocessor theory and practice, assembly language and interfacing. (As Flight Electronics will soon have a small robot arm connected to the MPF-1/88 it can even be used as part of an introduction to robotics and control.) There is even the possibility of using a number of MPF-1/88s connected to an IBM PC via the compatible expansion connector to store and down load software.

There is the question of whether the 8088 is a good microprocessor to use to introduce the fundamentals. Although it is not exactly the best thought out microprocessor ever built, it is a realistic example of what is to be found in current equipment. If you teach or want to learn about the 8088 then the MPF-1/88 is a good choice of hardware.

The MPF-1/88 costs £325 and is available from Flight Electronics Ltd, Quayside Road, Bitterne Manor, Southampton. Tel: (0703) 34033/27721.

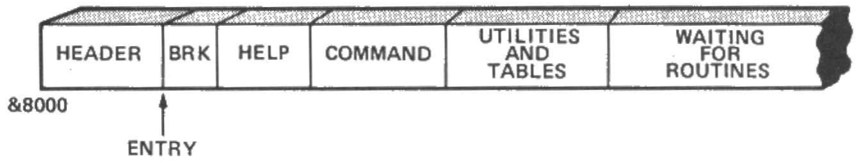
DATA MPF-1/88

Processor	16-bit 8088 4.77MHz
RAM	4K expandable to 24K
ROM	16K expandable to 48K
Display	20 chr 2 line LCD
Keyboard	59 key full-travel
Interfaces	16-pin Centronics Cassette interface 62-pin card edge connector.
Price	£325



ROM DESIGNER

If you want to develop your own ROMs, Mike Williams has already done all the hard work.



ROM MEMORY MAP

The rapidly falling price of 8K RAM chips (currently around £10) has persuaded more and more BBC micro owners to invest in a sideways RAM board. Manufacturers will suggest that the purpose of the boards is ROM development – but not a few are bought with the sole aim of downloading pirated ROMs from disk. However it's a lot more fun to create your own ROM and include on it your favourite routines, your own function keys, and to personalise your computer with such things as an individual power-up message.

To write ROMs you must have some understanding of the way a ROM is arranged so that it can obey *commands, and, for an easy life, you need some assembler routines.

The structure of any ROM falls into four main sections:

Header around 100 bytes giving the ROM title and some other legal information. This forms the gateway into the ROM.

Help routines that enable the ROM to reply to *HELP by listing the available commands.

Command routines to test whether a requested * command is one of yours, and if it is then sends the system off to the right place.

Routines the actual programs which carry out the * commands.

The first three sections form the main framework of any ROM. Once you have them sorted out it is just a matter of adding the actual routines for any specific ROM. This article describes and explains this essential framework. A subsequent article adds some routines which you can include in your own ROM and gives advice on converting machine code programs so that they will work in ROM.

The first section of the ROM Designer program is given in **Listing 1.** It starts by setting up some variables and then calls the header procedure. The function of this procedure is to place, right at the start of the ROM, the information which is often referred to as the ROM "signature". The details are described in the Advanced User's Guide and need not concern you as the header procedure does most of the work. You merely decide on:

- The ROM title which you insert in line 380
- The copywrite string to protect your handiwork. This is typically your name and the current date. Insert this into line 490.
- Finally, when the program runs, you will be asked for a version number. This helps to keep track of the ROM as it develops.

After typing in **Listing 1** (and saving it) you are welcome to RUN it and thereby to insert a legal header into your sideways RAM. However, pressing BREAK or entering any *command will cause the machine to lock up. Then I'm afraid that you will have to switch the computer off, then on again. The machine locks up because when BREAK is pressed the system jumps to the address, entry%, in bytes &B004,5 (lines 530, 540). But there isn't anything there yet!

To make the ROM *do* something, first type in **Listing 2.** This gets us into the ROM proper. The operating system enters a ROM for one of three reasons:

BREAK pressed.

Accumulator contains 3

*HELP entered.

Accumulator contains 9

*COMMAND entered.

Accumulator contains 4

So our ROM must first check to see what is in the accumulator.

LISTING 1.

```
100 REM ******************
110 REM *  ROM DESIGNER *
120 REM *       by       *
130 REM *  M.E.Williams *
140 REM *  (C) MEWsoft   *
150 REM ******************
160 :
170 rom_start=&8000
180 osasci=&FFE3
190 osnewl=&FFE7
200 osbyte=&FFF4
210 oswrch=&FFEE
220 text_pointer=&50
230 dot_length=&51
240 jump_lo=&52
250 jump_hi=&53
260 PROCheader
270 PROCassemble
280 END
290 :
300 DEF PROCheader
310 rom%=rom_start
320 CLS
330 !rom%=&4C000000 : REM 4C = JMP.  See lines 520,540
340 rom%?6=&82 : REM ROM type.
350 rom%?8=0   : REM version number, modify if necessary.
360 :
370 rom%=rom_start+9
380 Rom_title$="Mike's Rom "  :REM please modify
390 PROCstring(Rom_title$) : REM poke in title and update rom%
400 :
410 INPUT"Version number",A$
420 PROCstring(A$)
430 :
440 copy_offset=(rom%-rom_start-1)
450 rom_start?7=copy_offset
460 PROCstring("(C)")
470 :
480 rom%=rom%-1                         :REM overwrite the 0 after (C)
490 copyright$="Captain Pugwash" :REM modify as necessary
500 PROCstring(copyright$)
510 :
520 entry%=rom%   :REM this is the entry point to the Rom
530 rom_start?4=entry% MOD 256
540 rom_start?5=entry% DIV 256
550 ENDPROC
560 :
570 DEF PROCstring(A$)
580 $rom%=A$
590 rom%=rom%+LENA$
600 ?rom%=0:rom%=rom%+1
610 ENDPROC
```

If it is a "3" then the system is sent to the Break routine. Many ROMs place a routine here that prints out the ROM's name to remind you of its presence. It may also set vectors or flags. It could be made to print your name and address and even to ask for a password to prevent illicit games playing!

Listing 2 is mainly concerned with making an appropriate response to *HELP. If a *HELP alone has been entered the routine rom_title_print does just what it says.

If *HELP is followed, not by a carriage return but by a character, then it may be that the user wants some help with our ROM. In that case the help routine has to test whether the string of characters following *HELP matches the ROM name. This is the job performed by check_help_string. Since your ROM title might be fairly long it allows for the usual "." abreviation.

If the string matches, and help is required from your ROM, then the routine print_command prints out the list of commands, and perhaps their syntax, which is stored at help_table. This table is set up by lines 2610-2670. It is up to you how much help to print at this stage. But the routine as

LISTING 2.

```
 630 DEF PROCassemble
 640 FOR PASS=0 TO 3 STEP 3
 650   P%=entry%
 660   [ OPT PASS
 670   PHP
 680   CMP #3:BEQ break
 690   CMP #9:BEQ help
 700   CMP #4:BEQ command
 710   .not_me
 720   PLP
 730   RTS
 740   \
 750   .break
 760   PHA:TYA:PHA:TXA:PHA
 770   \  here we put action on break
 780   JMP exit_rom
 790   \
 800   .help
 810   PHA:TYA:PHA:TXA:PHA
 820   LDX #0
 830   LDA (&F2),Y          \ get character following HELP
 840   CMP #&0D
 850   BEQ rom_title_print \ just HELP
 860   :
 870   .check_help_string
 880   JSR strip_sp   \ get next character
 890   BEQ cr_1        \ carriage return pressed
 900   CMP #ASC(".")
 910   BEQ rom_title_print  \ usual . abreviation
 920   CMP rom_start+9,X    \ compare with the Rom title
 930   BNE exit_rom
 940   INX:INY
 950   BNE check_help_string
 960   :
 970   .cr_1
 980   CPX #LEN(Rom_title$)
 990   BNE exit_rom
1000   \
1010   .rom_title_print
1020   JSR osnewl
1030   LDY #0
1040   .title_print
1050   LDA rom_start+9,Y
1060   JSR osasci
1070   INY                  \ have we done title
1080   CPY #(copy_offset-9) \ plus version number?
1090   BNE title_print
1100   JSR osnewl
1110   CPX #0   \ X=0 if just *HELP
1120   BEQ exit_rom
1130   :
1140   .print_commands
1150   LDY #0
1160   .p_c1
1170   JSR osnewl
1180   LDA #32
1190   JSR oswrch \ indent commands 2 spaces right
1200   .p_c2
1210   JSR oswrch
1220   INY
1230   LDA help_table,Y
1240   BEQ p_c1  \ found 0 at end of each command name
1250   CMP #9    \ end of command list?
1260   BNE p_c2
1270   JSR osnewl
1280   .exit_rom
1290   PLA:TAX:PLA:TAY:PLA:PLP
1300   RTS
1310   \
1320  .command NOP
2050   \
2060   ]
2070   P%=rom_start+&400: REM leave plenty of room
2080   REM   Here are general purpose routines
2090   :
2100   [ OPT PASS
2110   .message
2120   PLA
2130   STA &8E
2140   PLA
2150   STA &8F
2160   JSR mess1
2170   LDA &8F
2180   PHA
2190   LDA &8E
2200   PHA
2210   RTS
2220   .mess1
2230   JSR increment
2240   LDY #0
2250   .mess_loop
2260   LDA (&8E),Y
2270   BEQ mess_done
2280   JSR osasci
2290   JSR increment
2300   BNE mess_loop
2310   .mess_done
2320   RTS
2330   .increment
2340   INC &8E
2350   BNE m_jmp
2360   INC &8F
2370   .m_jmp
2380   RTS
2390   \
2400   \
2410   .strip_sp
2420   LDA (&F2),Y
2430   CMP #32:BEQ get_nxt
2440   CMP #&0D:RTS
2450   .get_nxt
2460   INY:BNE strip_sp
2470   \
2480   \
2490   .command_table
2500:
2570   .help_table BRK
2580   ]
2590   NEXT PASS
2600 rom%=P%
2610 PROCstring("FNkeys")
2620 PROCstring("WWkeys")
2630 PROCstring("Dump")
2640 PROCstring("Popup")
2650 PROCstring("Others")
2660 PROCstring("Etc")
2670 ?rom%=9
2680 ENDPROC
2690 :
2700 DEFFNmessage(A$)
2710 message$=CHR$13+A$+CHR$13
2720 $P%=message$
2730 P%=P%+LEN(message$)+1
2740 ?(P%-1)=0
2750 =PASS
2760 :
2770 DEFFNequs(A$)
2780 $P%=A$
2790 P%=P%+LEN(A$)
2800 =PASS
2810 :
2820 DEFFNequb(byte)
2830 ?P%=byte
2840 P%=P%+1
2850 =PASS
2860 :
2870 DEFFNequw(word)
2880 ?P%=word AND &FF
2890 P%?1=word DIV 256
2900 P%=P%+2
2910 =PASS
```

LISTING 3.

```
 1320    .command
 1330    PHA:TYA:PHA:TXA:PHA
 1340    STY text_pointer
 1350    LDX #0
 1360    LDA command_table,X
 1370    .check_table
 1380    STA dot_length
 1390    INX:LDA command_table,X
 1400    STA jump_lo
 1410    INX:LDA command_table,X
 1420    STA jump_hi
 1430    INX
 1440    LDY text_pointer
 1450    .nxt_ltr
 1460    LDA command_table,X
 1470    BEQ com_end    \ successfully reached command end
 1480    DEC dot_length
 1490    LDA (&F2),Y
 1500    AND #&DF        \ accept lower case
 1510    CMP command_table,X
 1520    BNE not_command
 1530    INY:INX
 1540    BNE nxt_ltr    \ always
 1550    .com_end
 1560    LDA (&F2),Y
 1570    CMP #&41       \ next character alphabetic?
 1580    BCS get_past   \ if it is, its not our command
 1590    JMP do_it
 1600    .not_command
 1610    CMP #&0E       \ is it a dot?
 1620    BNE get_past
 1630    BIT dot_length     \ done minimum required length?
 1640    BPL get_past
 1650    INY:JMP do_it
 1660    :
 1670    .get_past
 1680    INX
 1690    LDA command_table,X
 1700    BEQ get_past \ unly the terminating 0
 1710    CMP #&FF
 1720    BEQ exit_rom
 1730    CMP #&09      \ keep going till we
 1740    BCS get_past \ reach the dot number
 1750    JMP check_table
 1760    \
 1770    .do_it
 1780    JMP (jump_lo)
 1790    \
 1800    .done_rom
 1810    PLA:TAX:PLA:TAY:PLA
 1820    LDA #0
 1830    PLP
 1840    RTS
 1850    \           Now follow the command entry points
 1860    .command1
 1870    :\ Routines to go here
 1880    JMP done_rom
 1890    .command2
 1900    :
 1910    JMP done_rom
 1920    .command3
 1930    :
 1940    JMP done_rom
 1950    .command4
 1960    :
 1970    JMP done_rom
 1980    .command5
 1990    :
 2000    JMP done_rom
 2010    .command6
 2020    :
 2030    JMP done_rom
 2040    :


 2500    OPT FNequb(2):OPT FNequw(command1):OPT FNequs("FNKEYS"):OPT
FNequb(0)
 2510    OPT FNequb(2):OPT FNequw(command2):OPT FNequs("WWKEYS"):OPT
FNequb(0)
 2520    OPT FNequb(2):OPT FNequw(command3):OPT FNequs("DUMP"):OPT
FNequb(0)
 2530    OPT FNequb(2):OPT FNequw(command4):OPT FNequs("POPUP"):OPT
FNequb(0)
 2540    OPT FNequb(2):OPT FNequw(command5):OPT FNequs("OTHERS"):OPT
FNequb(0)
 2550    OPT FNequb(2):OPT FNequw(command6):OPT FNequs("ETC"):OPT
FNequb(0)
 2560    OPT FNequb(&FF)
```

written will only cope with a total help_table length of 256 bytes. That is more than enough to cope with command names and syntax.

The program so far, listings 1 and 2 combined, can be RUN and the computer should then respond to *HELP and to * HELP <your ROM name>. But please remember that errors in typing in the program will possibly result in the machine crashing beyond' the reach of BREAK accompanied by muttered oaths and curses. So save the program before running it.

The next part of the designer has to obey *COMMANDS. When the operating system encounters *<anything> it places a 4 in the accumulator and then goes to each ROM asking, in effect, "did you call?" The routines in **Listing 3** enable the computer to check each of the ROM's commands in turn to see if they match the *command given. The list of commands is stored at command_table. It has the following structure for each command:

<dot number> <command lo & hi>
<command name> <0>
The end of the table is signified by a single &FF byte.

The dot number is concerned with the ability to shorten a command name using a dot, ie '.'

for example *H. for help

To avoid clashes between commands with similar names it is usually necessary to have a minimum of 2 or 3 letters. This minimum number is what I have called the "dot number".

<command lo & hi> is 2 byte giving the low byte and high byte of the actual routine which carries out the command. To keep life organised these routines are entered at the *command points in lines 1860-2010.

Next comes the command name itself, terminated by a 0 byte. The command_table is itself terminated by a byte of &FF.

The command table is constructed for you by Lines 2500-2560. (Lucky owners of BasicII do not need to use the OPT FN technique.) The operation of the command routine is fiddly to explain. If you are really interested the copious assembler comments should help.

The section following the command entry points, at rom_start%+&400, is a good place to put general purpose routines which might be used by many of your ROM routines. One such is the well known message printing routine given. Other routines could handle data entry, ASCII to hex conversion and so on. By placing them here you know exactly where they are so that they can easily be called by your command routines.

The memory map of the ROM so far is shown in **Figure 1.**

It only remains to place some useful routines in the ROM space waiting for them; insert a JSR at the command entry points and we have a working ROM. In the next article I will supply you with some ready made routines and will offer advice on converting your machine code programs so that they work in ROM.

Focus on QL software

John Banks takes a further look at new additions to the QL software scene – and finds a very mixed bag.

MonQL

Hisoft
Price: £19.95

The manual supplied with this software describes it as a machine code monitor/debugger, but disassembler might be a more familiar term (anyone not familiar with either description is probably not going to find a use for it anyway).

Hisoft's MonQL is aimed directly at machine code programmers or anyone involved in the manipulation of object files. It can be placed in two distinct areas within the QL's memory and once loaded is accessed via three SuperBasic extensions – MONQL, MONEXEC and MONEXEC_W.

Once entered, MonQL employs a separate window split into three sections. These display the current contents of the CPU's registers (including PC, Status and Supervisor – 87'), the 24 bytes around the Memory Pointer and a four-instruction disassembly.

The package encompasses all the usual commands associated with disassemblers but unfortunately does not include an assembler so that any later editing must be done separately.

MONQL Commands

G	– Get a sequence (search)
I	– Intelligent copy
J	– Job control
M	– Modify memory pointer
N	– find Next match (after G)
P	– Print disassembly to printer or screen
Q	– Quick disassembly to screen
R	– Register alter
T	– Temporarily update memory pointer
V	– display Various bases
W	– fill memory With byte
X	– eXamine memory
↑A	– put breakpoint after current instruction then execute it
↑B	– set/reset Breakpoint
↑K	– Kill all breakpoints
↑M	– change screen Mode
↑P	– redefine Printer device
↑Q	– Quit to BASIC
↑R	– Return to whatever called MONQL
↑X	– eXamine memory onto printer
↑Z	– single step
0-9,A-F	– enter data into memory
"	– enter ASCII string into memory
ENTER	– increase memory pointer by a word
SHIFT ENTER	– increase memory pointer by a long word
–	– decrease memory pointer by a word
SHIFT	– decrease memory pointer by a long word
←	– decrease memory pointer by a byte
→	– increase memory pointer by a byte
↑	– increase memory pointer by a word
↓	– decrease memory pointer by a word
↑F1	– toggle register display between all registers & just PC/SR
↑F2	– toggle memory display on and off
↑F3	– toggle disassembly between 1 and 4 instructions
↑F4	– move window around screen

Executive Adventure

Intersoft
Price: POA

Unlike most games of this genre, Executive Adventure contains no hidden treasures to collect but involves a series of tasks to complete in the quest for executive glory.

This is one of those games where speed is of little importance – a keen mind will win the day and bring about the noviciate adventurer's transformation from lowly tramp to Chairman of the board (the aim of the game).

The first problem to be overcome is how to get out of the maze that begins every game. Asking for hints merely gains the response "Pockets may need investigating" and once you've found the coin then it's back to aimless wandering.

But don't give up: perseverence reaps rewards and our intrepid hero is soon confronted by a fountain. It doesn't take an Einstein to work out what to do with the coin next and soon we're out of the maze (sorry – forest!) and on the High Street.

Now the standard adventure pattern emerges, or rather the pattern recognisable to anyone who has played a Richard Shepherd adventure on, say, the Commodore 64. Not that this is a criticism of the game but rather an observation all too familiar with QL games – 'nothing new here'.

Without revealing too much, your progress through a series of not unfamiliar locations ('Harridges' or 'Bark-West bank' for instance), and a diverse assortment of cynically humerous encounters – like the reference to J. R. Ewing when you are inadvertently shot whilst trying to achieve the ultimate accolade, promotion to company Chairman.

After all that, the end of the game is an anticlimax (and a cheeky advert for Intersoft), but to Intersoft's credit, this is mainly due to it being such an involved affair to get there and reminiscent of Melbourne Houses's 'Hampstead'. In fact, the majority of adventurers are likely to require several weeks before they reach the top.

QL Bank Account

Cenprime Software
Price: POA

Of all the software currently around for the QL, the vast majority can be pigeon-holed into one of two categories – business or pleasure. QL Bank Account does not fit snugly into either of these – it's definitely not a game, but neither is it a package for small businesses. Instead it's one of a limited number of home/professional packages which hopefully will become more abundant as the machine's user base grows.

QL Bank Account is a computerised statement checker which allows you to keep tabs on regular (eg. standing orders) and irregular (eg. cheques) expenditure as well as providing extensive summary and reporting facilities on the status of your account.

The program caters for up to 17 different groups of payments (eg. gas, electricity etc) and 20 standing orders, each independent of the others. Monthly, quarterly or annual payments to be made every four weeks – a significant ommission.

Entering data couldn't be easier as the program is menu driven throughout and full instructions are provided on each new screen. And if that isn't enough all the instructions are duplicated and expanded in the accompanying 24-page manual.

Once an initial or preliminary file is set up, all that remains is to enter any subsequent transactions as they occur and the program will print (on the screen or a suitable printer) an overall balance. Alternatively, the whole file or selected entries can be grouped and printed out (though make sure a printer is connectred as the program is curiously unable to recover from any 'device not present' errors – the only other criticism that can be levelled at the package).

Considering the inferior quality of a fair proportion of QL software, Cenprime would appear to be forging their own path with QL Bank Account – and in the right direction!

QL APPLICATIONS SOFTWARE

Company	Contact	Products
Sinclair Research	0376 686100	QL Cash Trader, QL Entrepreneur, QL Project Planner, QL Decision Maker,
Cenprime	0203 686162	QL Bank Account
Adder	0223 277050	Q-Doctor, Assembler
Computer One	0223 86216	Pascal, Forth, Assembler, Monitor, Typing Tutor
Co-op Soft	0272 22223	Civil/Structural Engineering
Dialog Software	01 500 2386	Transact Book-keeping package
Digital Precision	01 527 5493	Games Designer, Sprite Generator, Dissassembler + Monitor
GST	0954 81991	Q Jump, Toolkit, Assembler
Harcourt	0276 686100	Touch 'n go
Hisoft	0582 696421	Mon-QL
Metacomco	0272 428781	Assembler, BCPL, Lisp
MicroAPL	01 622 0395	
Portfolio	PO Box 15, LONDON, SW11	Stockmarket Manager
Positron	0554 759624	Hi-res screen dump
Q-Soft	01 449 7417	Agenda Desk Diary
Quest	04215 66488	Business accounts
Super Plant	097 423223	Shrub Planner, House plant planner
Tasman	0532 438301	Tascopy Screen Copier
TDI Software	0272 742796	USCD Pascal, USCD Fortran, Advanced Toolkit
TR Systems	093 924 621	QL payroll

QL GAMES

Company	Contact	Products
Digital Precision	01 527 5494	Backgammon
Eidersoft	01 478 1291	QL Art
Games Workshop	01 965 3713	D-Day
Printerland	0484 513105	Psion Chess
Shadow Games	0296 686100	Area Radar Controller
Talent	041 552 2128	ZKUL, WEST, GraphiQL

Zappit

Quest
Price: £9.95

The title of this program might lead the uninitiated to assume it's a shoot-em-up game – nothing could be further from the truth. This is not a game at all but a "powerful Sprite designer".

Sprite designer it may be, but powerful is another matter – a better adjective might be 'straightforward'. In no way does Zappit match up to the likes of Talent's Graphiql. It is not intended as a comprehensive graphics package, nor does it include a detailed manual on how to implement the commands, but where Zappit does score is the ease with which Sprites can be created and manipulated.

When the program is first booted up, there are two procedures to run through before any sprites can be used. The first specifies the number of sprites to create and how they will move around the screen. The second permits the user to set the colour and shape (à la MacPaint) for each individual Sprite.

Once these two sets of parameters have been fixed, two files are saved onto microdrive and the Sprites can now be accessed from any SuperBasic program after following the initialisation procedure detailed on the instruction sheet.

In effect, Zappit extends the SuperBasic command vocabulary to include seven extra Sprite commands and four extra functions. These include SMOVE (pixel-based movement), CLR (a sprite version of CLS), SETDIR (amend directional characteristics) and SETJUMP (pixel jump height). Of greater value are the four functions, which provide a means of detecting collisions, as well as direction and position information.

Quest has hit upon a quick and easy way to extend the QL's graphics capabilities and more than adequately fills the gap left by SuperBasic's lack of UDG commands.

Fantasia Adventure

S&B Software
Price: POA

The form of the game, though described within a framework of espionage and hostile territories, shows similarities to the Melbourne House Classic Adventure with its caves, tunnels and passages. Indeed, this is a game for purist adventurers and quite an extensive one too.

In addition to the usual smattering of Knights, swords and daggers, you are confronted by bolshy guards with assorted explosive miscellanea. Get caught wearing the wrong clothing or without the right headgear and it's curtains.

A small criticism here is that death does seem to hinder your progress with frustrating regularity and then it's back to the beginning! It's also difficult at times to remember the plot of Spy in a Hostile Country.

Blackjack

Quest
Price: £18.95

It seems as though Quest has been in the QL market almost as long as Sinclair, but it's only recently that this company has emerged on the software scene with some impressively packaged games (though this term is used in the loose sense of the word).

Blackjack (AKA Pontoon) is a prime example of Quest's talent for professional finishing and deserves praise from the word go. Upon loading, anxious players are reminded of the importance of making back-up copies from the master drive and the accompanying instruction sheet give full details of how to do this. The game can then be played without fear of losing everything when the QL inevitably crashes.

All the standard Blackjack variations are covered – splitting, five card tricks, 'Blackjack' – though that's about the extent of the game and as there are no variations (skill levels for instance) Quest's Blackjack would appear to be more of a five minute wonder than anything really challenging.

Morse Tutor

WD Software
Price: £6.00

This aid to learning Morse Code has obviously been designed with a view to practicality rather than fancy graphics and presentation. It is supplied with a single sheet of instructions and in the original microdrive packaging, but then it is less expensive than most QL software at £6.

Aimed at the complete beginner, Morse Tutor can cope with slow (one character at a time) or fast speeds (up to 18 words per minute) and contains a test library of 100 sentences. Output from the program can be duplicated on a printer and at the same time the pitch of the beep from inside the QL can be altered to suit your tastes.

Learning Morse this way is effective and you can soon work up quite a speed (or at least it seems that way). The instructions recommend occasional intellectual stretching to 15 wpm, but just being able to translate a whole sentence is fairly rewarding.

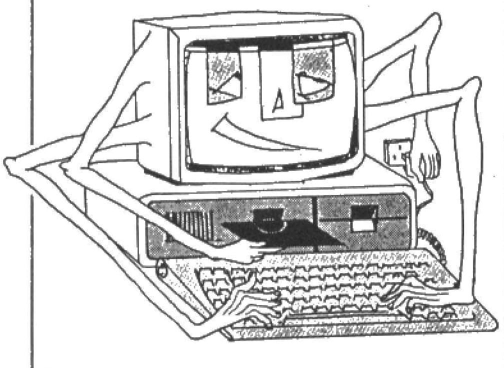
As a piece of exciting software, then, Morse Tutor does not really get off the ground, but as a cheap and cheerful 'teach yourself' aid to the learning of a specialist skill, it's a high flyer.

SUPPLIERS

Intersoft 7 Richmond Road, Exeter, Devon.
Quest 04215 66488
S&B Software 20 St. Nicholas Street, Diss, Norfolk.
Cenprime Software 10 Castle Street, Rugby CY21 2TP
Hisoft 0793 26616
WD Software 0534 81392

Easy living with an operating system

An operating system (should) provide an easy pathway to a computer's full facilities, according to Mike James.



There is no more misunderstood and yet important piece of software than an operating system – it can make a mediocre machine look good and an excellent machine look appalling. Until now there has been a general lack of interest in operating systems because the traditional range of personal computers have used BASIC as their 'front line' software. That is, when first switched on a machine like the Spectrum, the BBC Micro or the Commodore 64 works in BASIC and any operating system-like features that may be required are obtained by extending BASIC to include extra commands. Machines such as the ACT F1e, the Sanyo 550 and the IBM PC (and compatibles) have moved away from this one language approach however and use an operating system. This move away from the BASIC machine to a versatile multi-language, multi-application, operating system based machine is bound to continue as our personal computers become more powerful.

What is an OS?

The real question that troubles most users is "what is an operating system for?" The purpose of a language such as BASIC or an applications package such as a wordprocessor is quite clear but an operating system doesn't seem to 'do' anything specific. The main purpose of an operating system is to enhance the appearance of the machine to the user, both in terms of efficiency and ease of use; to direct service to the user, eg it must provide commands such as CAT, COPY and to service applications programs running 'under' the operating system, eg it must handle requests to read a file from disk or print on a printer.

The ease of use of an operating system is mainly influenced by the range and quality of the direct services it provides. Two operating systems can provide exactly the same range of user services (ie anything you can do on one you can do on the other) but one may be easy to use and one difficult. This appraisal of what is difficult is clearly subjective – people like different things. One can argue though that the ease of use of an operating system (in fact any software) relates to how logical its commands are. To some extent this can be measured by how long it takes to learn how to use the operating system. If its commands are logical then there will be a uniform pattern of use among all the commands. In other words, once one command has been mastered, the rest are very similar. This question of logicality of commands is not the only influence on ease of use. There is also the degree to which the commands seem natural although this is a much more difficult quantity to describe let alone measure. How natural an operating system seems is entirely a matter for personal judgment. For example to make a second copy of a file using CP/M (one of the oldest micro operating systems) the command structure is:

```
PIP NEWFILE=OLDFILE
```

but using MSDOS (used on the IBM PC) the equivalent command is:

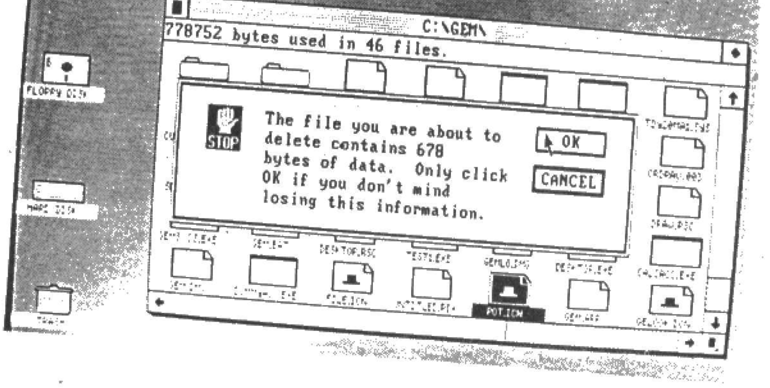
```
COPY OLDFILE NEWFILE
```

some people swear by the first version and some the second. Notice that the difference between the two is not just confined to the use of the words PIP and COPY – the order of the files is different.

Two operating systems can differ not only in the case of use of their commands, but in the range of commands they offer. A very simple, easy-to-use operating system may achieve this 'ease of use' by being restrictive. That is, it may be easy for a newcomer to begin but a persistent user will quickly out-grow the system and have to start all over again. To compare two operating systems from the point of view of ease of use it is vital to identify the range of commands and facilities that are being used to compare them. An advanced, full-featured operating system may be difficult to learn if the full advantage of all its facilities is to be appreciated, but to say that it is difficult for a beginner to get started on is a very different matter. (It is always more difficult to do difficult things!)

A second broad area that influences an operating system's ease of use is the quality of its English! Operating system commands are the way that the user 'talks to the operating system'. The operating system also talks back via user messages

GEM on the IBM PC: no more than a front end for PC-DOS but a lot easier to use.



C:\GEM\

778752 bytes used in 46 files.

The file you are about to delete contains 678 bytes of data. Only click OK if you don't mind losing this information.

giving information about what is going on and the system status. A second less obvious form of communication from the operating system is its manual. Here lies the only opportunity that the operating system's designers have of communicating the worth of their creation. The trouble is that most operating system writers are better at program code than English and hence the resulting quality of operating system messages and manuals leave much to be desired (and this is one area where it is possible to be fairly definite – there are no good operating system manuals, just degrees of awfulness!)

The most important type of message that an operating system sends to a user is an error message and here too most operating systems fail to communicate just what the trouble is. True, after a while any sort of incomprehensible message begins to make sense just by virtue of the circumstances in which it tends to appear – but this is not really good enough! As well as informing a user of an error it is also important that reasonable corrective action is applied following the error – or at least a choice of actions should be presented to the user. Most operating systems do not do the right thing when it comes to an error. The most common response is to panic and just give up – if this involves the loss of a file of precious irretrievable data then the only person who seems to care is the user!

'an OS can make a good machine look mediocre or a bad machine look good'

Using the machine

As well as affecting the ease of use of a machine, an operating system can improve or reduce the overall efficiency of a machine. There are three main areas where this can be so.

MEMORY UTILISATION

An operating system is sometimes described as a program that occupies memory whether the user wants it to or not! Clearly there is always a cost in terms of available memory in having an operating system installed. The more complicated an operating system, ie the larger the range of facilities it provides, the more memory it needs. (Some operating systems are clearly wasteful in their memory use.) An operating system can however increase the amount of memory available to an applications program by using 'memory management' techniques. In micros these either involve the use of 'bank switching' or using disk storage as a slow extension to main memory.

PROCESSOR UTILISATION

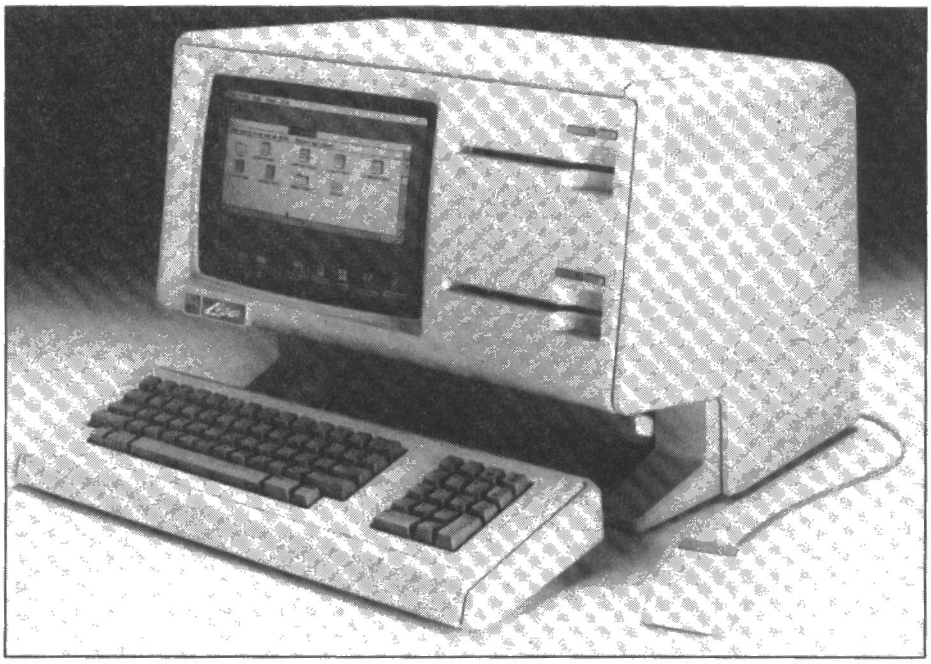
In any machine the CPU is not always able to find something to do. For example, if it is running a program that reads and writes to disk a lot, it can find itself 'hanging around' waiting for the disk unit to complete a read or a write for more time than it spends doing useful computing. An operating system can make use of this wasted time by 'multi-programming'. Whenever the CPU is idle the operating system can redirect its efforts to a different task. This means that a single user can give the machine more than one thing to do at a time – this is 'multi tasking'. A limited form of multi-tasking that is often encountered in micro operating systems is 'printer spooling'. In this case only two programs are run together, an application program of the user's choice and a system program that transfers data from a disk file to the printer. Thus printer spooling allows a machine to print a file and get on with another task at the same time and is a highly desirable feature of an operating system. A second possibility is that the machine is more than powerful enough to handle a single user's demands. For example, if a machine is being used as a wordprocessor even the fastest typist cannot supply characters at a rate that comes anywhere near the maximum possible rate that the machine can handle. In this case it makes sense to divide the machine's attention (and resources) between a number of users – this is called 'multi-user'. (There is a counter argument to multi-user operation, however, that maintains that as hardware is so cheap (and becoming cheaper all the time) it is not worth the extra trouble).

DISK UTILISATION

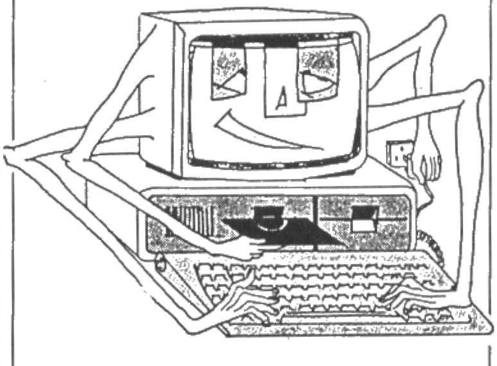
Every computer user expects to be able to use files of information stored on some device. What is less obvious is that files are entirely the creation of the operating system. (It is worth pointing out that this mirrors the creation of variables by languages such as BASIC. That is computer memory is not split up into named areas of storage such as COUNT or TOTAL – these are entirely the invention of BASIC.) A disk drive, for example, can only store and retrieve data in chunks known as sectors. A particular sector is identified by two numbers (itsd track and sector numbers). The operating system takes these raw units of storage and collects them together into larger entities referred to by names – ie into named files. The characteristics of the files available on any machine is thus wholly the product of the operating system. The disk drive sets the maximum speed that data can be read or written to but the operating system defines the speed at which files can be read and written.

There are clearly other facilities and services that operating systems may or may not offer. However most of these fall into the general area of I/O. For example, a good operating system will allow programs to read and write data without worrying about what sort of device will be supplying or accepting the data. This is called 'device independence' and it simplifies the production and use of applications programs considerably.

Rather than carrying on listing abstract features and advantages of operating systems it is better to describe some of the more important available for microcomputers.



The Apple Lisa was the first machine to use an icon based, mouse driven OS. It was a marketing disaster, but proved to be a successful prototype for the Macintosh.



A guide to common operating systems

CP/M 80

This was the first major microcomputer operating system and it is still in use in vast numbers. It runs on almost any hardware using either an 8080 or a Z80 CPU, at least one disk drive and around 20K of memory. It is a single-user, single-task operating system that offers minimal features and support. CP/M is slow in handling disks, is legendary for its poor error messages and manuals but it does have a very wide following and a very wide selection of applications programs running under it. Various improved versions of CP/M are available from other vendors – these are often referred to as 'turbo-DOSs' and some of them are definitely worth the extra. An improved version of CP/M called CP/M plus can also be obtained. CP/M is still the standard operating system for 8080/Z80 systems using floppy disk drives.

CP/M 86

This is essentially CP/M 80 rewritten for machines that use the 8088 and 8086 CPU such as the IBM PC, the Apricot etc. It has never achieved the same level of success as CP/M 80 because IBM chose PC-DOS as its standard operating system and the rest of the world followed!

MPM V2

MPM is a multi-tasking, multi-user version of CP/M – it is available for both the Z80 and the 8086/8088 family of micros. Although Version One received a lot of criticism (because it didn't work!). Version Two has been better thought of. Even so MPM has hardly become popular and this might be because of its poor early start and the lack of power of the Z80 to run multi-tasking/user software.

Concurrent CP/M and Concurrent PCDOS

Concurrent CP/M (CCPM), and its incarnation for the IBM PC – Concurrent PCDOS, is probably the most interesting new operating system on the market. It is a multi-user, multi-tasking operating system but it is better to use it as just a multi-tasking operating system. It performs multi-tasking in a way that is more suitable for a personal computer than other operating systems. The user can switch between a number of 'virtual screens' each one running a different application independent of the rest. Thus it is possible to look something up in a data base while writing a letter without having to leave and reload either the data base or the word processor. Another advantage of CCPM is that it is compatible with MSDOS V1 and V2 at both the data and program level. That is it will read and write MSDOS files and run MSDOS programs. CCPM is suitable for any 8088/8086 machine but it does need rather a lot of memory to get the best from it (around 512K!) and a hard disk.

MSDOS V1 and PCDOS V1

MSDOS V1 is essentially the same thing as PCDOS V1: the only difference is that PCDOS is intended for use with the IBM PC. MSDOS V1 is single-user, single-tasking and is very similar to CP/M in its outward appearance although its internal workings are very different. Anyone used to CP/M could sit down and use MSDOS/PCDOS without noticing too much difference apart from one or two simplifications (ie there is a COPY command included as standard). As MSDOS is intended for 8088/8086 CPUs and is supplied with machines such as the ACT Apricot F1e and the range of IBM compatibles it is now the most commonly used operating system.

MSDOS V2 and PCDOS V2

Although the addition of a Version Two indicator usually means that there is very little difference from Version One, in this case it is important to realise that V1 and V2 are very different. MSDOS V2 and PCDOS V2 include many extra facilities over V1, including multiple disk directories, comprehensive printer spooling and I/O redirection. The rule is that MSDOS V1 is for a system that has a pair of floppy drives and around 64 to 128K of memory and V2 is really only necessary for systems with a hard disk and more than 128K of memory.

UCSD P system

The USCD P system started life as an implementation of the Pascal programming language but it has now grown to the point where it is important as an operating system offering a range of high level languages. It is slow and fairly restrictive in the range of facilities it offers but it is easy to use if you are developing programs. USCD P has :ound its way into the next generation of operating systems by offering multi-user/multi-tasking facilities – but for those not interested in Pascal program development it is a non-starter.

Smalltalk

This is one of the newest and most exciting ideas in operating systems. Instead of using commands, Smalltalk communicates via graphics. For example, it presents the user with a picture of a desk complete with in and out trays for filing operations. This is likely to be a good way to proceed for computer naive users but it does demand excellent graphics facilities. This graphics approach to operating systems has become known as 'iconic programming' and it has been taken up by other operating systems. It is important to realise however that iconic programming is not the only feature of Smalltalk that is new – it also introduces a whole new approach to computing called 'object programming' that other operating systems have not yet copied. Smalltalk is only now becoming available for the IBM PC and the Apple IIe and IIc.

The Mac operating system

This is essentially an icon based operating system borrowing much from Smalltalk. It is, or course, only available on the Apple Macintosh and so it's good and bad points are the good and bad points of the Mac!

GEM

GEM is not so much an operating system more of an add-on. Put simply it turns any of MS-DOS, PCDOS or Concurrent CP/M into an icon based operating system. The most notable use of GEM is in the new Atari machines which will bring iconic programming into the home computer price range.

Unix and Xenix

Unix is a full multi-tasking, multi-user operating system and Xenix is just an implementation of it for the IBM PC and compatibles. In essence it is a scaled down version of the sort of operating systems used on mainframes. It offers a good hierarchical filing system. It is though poor on user errors and is prone to intentional interference. It also uses a great deal of memory and is far from easy to use. It is available on a wide range of machines and in a wide range of versions from different suppliers. Unix is not so much an operating system more a group of very similar operating systems. It has always been said that Unix was the operating system of the future but the future still seems some way off!

a bunch of five

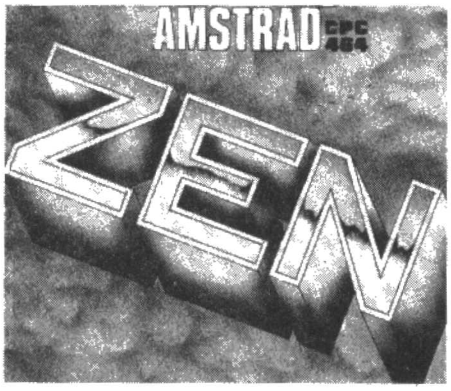
With its open architecture and five – count them! – assemblers, the Amstrad is a good computer for machine code virtuosos. Peter Green has been comparing products.

Amstrad has been very open in providing details of its operating system to machine code programmers; unlike some small-minded manufacturers, one of whom brought an injunction against a published disassembly of its ROM. Amstrad's more enlightened attitude deserves success, with no less than five companies now marketing assembler/monitors for the CPC464. The success has been achieved, but which one to buy?

The basic job of an assembler is to turn a source file of assembler mnenomics into the corresponding machine code. Its text editor should be almost as powerful as a word processor, so that the source text can be written easily. The assembly routines should have versatile expression evaluation and labelling so that any type of jump can be implemented easily. It should never, never corrupt your source of object code!

A monitor is essential to debug all but the simplest machine code programs. It should allow easy examination and manipulation of both memory and the processor's registers, be relocatable so that it can be used with programs wherever they are sited in memory, and disassemble code into assembler mnemonics. It must let your single-step through a piece of machine code, or at least stop the program at predetermined breakpoints. And again, you must be able to trust it not to corrupt things unexpectedly.

With this in mind, we can grade our five contenders.



ZEN

Cassette £19.95
Kuma 07357 4335

New versions of this assembler appear each time a Z80-based computer is released, usually within weeks of the launch. Conversion is quick because the basic program core remains the same: this makes it tried and tested, but dated. It also means the program isn't consistent with Amstrad Basic. For example, assembler mnemonics have to be entered in capitals to be recognised. Why? The Basic accepts upper or lower case keywords; so should the assembler. It's called user-friendliness, isn't it? Don't confuse the customers.

ZEN occupies a little over 6K of RAM, starting at 4000 hex. It is not relocatable, and to accommodate it you have to alter HIMEM yourself. Surely Kuma have heard of BASIC loaders?

With only 6K, ZEN is bound to be primitive. Like Devpac, commands are entered as single letters: 24 upper case and three lower case. Lines are always numbered from 1 in steps of 1. T lets you jump to a given line, U and D move the cursor up or down. You cannot edit a line, only ZAP it and retype it. ZAP will remove a block of lines, but uses an odd syntax: T1080, Z6, for example, instead of the more natural D(elete)1080-1085. This syntax is also used with P to print part of the text file.

Another oddity is in the assembly routine. Unless explicit request is made for object code by including a LOAD directive at the start of the text file, none is forthcoming. Assemblers normally assume you want code produced (which is usually the case) unless they are told otherwise.

At first glance the thick manual looks daunting, but most of it is taken up by the source code listing of ZEN itself. The actual instructions on using ZEN occupy a terse 10 pages. They could do with expansion: the GOTO command for running object code mentions setting up the Z80 registers with the values in the 'User Image', but not how to access this image.

ZEN is really starting to show its age, and needs a total facelift to compete. The Ford Model T was a bestseller in its day, but would you really want one in preference to an Escort?

DEVPAC

Cassette £21.95
Disk £26.95
HiSoft 0582 696421

Devpac, by HiSoft is two programs: GENA3, the editor/assembler, and MONA3, the disassembler/monitor. Both are relocatable within the range 1000 to 30000 decimal, and may be used together or independently. With both loaded, about half of the machine's user RAM is free for source text files and assembled object code. As with most of HiSoft's offerings for the CPC464, Devpac has its origins on the Spectrum.

GENA3 uses single-letter commands to implement its functions, followed by up to two numerical arguments and two string arguments. This restricts the possibilities and some mnemonics are a little forced: you PUT a text file onto tape or disk rather than SAVE it, because S changes the argument separator in command statements. You GET files rather than LOAD them, because L is for LIST. And Z , of course, lists the source code to the printer!

Like BASIC, source code lines are numbered, but the editor is nowhere near as good as Amstrad's. The old TRS80 style line editor is used, with its own set of single key commands. The cursor keys don't work, nor do you have BASIC's COPY facility. The delete and space keys step a pointer through the line editor buffer; pressing K 'kills' letters, otherwise press I to enter insert mode and then use the delete key, UK.

The maximum line length is 80 characters, according to the manual – but don't use all 80! I tried to edit an 80-character line with the X command (jump to end of line and enter Insert mode) followed by Delete. The Amstrad performed a total reset. Of course, it was over an hour since I'd last saved my source code.

Also, Devpac won't assemble a program which contains 80-column lines. If you try, the first two lines in the source file are corrupted, and all subsequent jumps in the object code have the wrong offsets. Apparently these bugs are known. They are not going to be fixed. Would you buy an assembler from this man . . .?

MONA3 has a comprehensive 'front panel', which shows the values in each Z80 register and memory contents of the bytes indexed by the register pairs, the state of the flags, the op code for the byte currently addressed by the memory pointer and a display of 32 bytes centred on the memory pointer.

Considering this is the official Amsoft assembler and monitor, it is sad that it doesn't come up to scratch. As a commercial programmer, I couldn't trust my livelihood to a program with such blatant bugs and inefficient implementation. I was given my copy free, but I paid out money for a rival product. And I know that one of the other products mentioned here was actually written by programmers to avoid using Devpac.

ZAPP

ZAPP

Cassette £14.95
Hewson Consultants 0235 832939

Or, Z80 Assembly Programming Package. ZAPP is available only on cassette. However, Hewson's manual tells disk users how to make a backup copy on disk, which is public-spirited of them (are you listening, Acornsoft?).

Hewson has dispensed entirely with single letter entry: instead, this system looks like a cross between the BBC OS and the Spectrum.

The text editor is fairly rudimentary but adequate. It is of the auto-dynamic line number type, meaning that every time a line is entered or deleted, the whole file is renumbered from 1 in steps of 1. I can't see the point in this: if the file is constantly being renumbered as you alter it, the numbering isn't much help for finding your way round. Luckily there are other methods available. You can scroll up and down through the listing by line or screenful, or find a label or operand by using the string search facility.

What I particularly dislike is that, when editing an existing line, pressing CLR erases the whole line. Since the BASIC editor only erases a single letter, editing habits have to be curbed or lines will be lost. Also, most assemblers insist that a space is left between a label and the following mnemonic: ZAPP throws the line back as an error unless you leave the space out!

The unique quality of ZAPP is that it's a one-pass assembler. Normally this would rule it out for serious work because forward labelling would be impossible. Hewson has solved this problem be constructing a table of labels as ZAPP assembles the code. At the end of its single pass. when it has located all the labels, it can use the table to go back and fill in the blanks instead of having to make an entire second pass through the source text. The only problems are first, that forward labels cannot be used in expressions, and second, that the table is of the 'linked list' type, with the link being only a single byte. Thus chaining errors can occur if more than 255 bytes of object code separate two labels, and here it is necessary to insert dummy labels to 'bridge the gap'.

The monitor permits single-stepping through code by pressing ENTER, displaying the Z80 registers and flags after each op code is performed. The manual doesn't explain how to get out of the monitor, but I will: just press ESC.

This is an interesting product with some unusual features, but could really do with a bit more detail in some parts of the manual.

MAXAM

Cassette £13.50
Disk £26.90
ROM £59.90
Arnor Ltd 01 653 1483

This piece of software should be held up as an example of what can be done by programmers who care. It is brilliant, I paid money for it rather than use my reviewer's freebies, and I wouldn't be without it. (In fact I've written a routine that lets the Amstrad produce Sinclair-compatible binary tapes, so I can develop programs for the Spectrum with Maxam and transfer them across.) It occupies virtually the same amount of memory as Devpac, but there the comparison ends.

Maxam is available on cassette, disk and external ROM; Arnor is the first and only company to make use of this major facility that the Amstrad offers, and is to be commended for its implementation. I hope it leads to an explosion in ROM software such as BBC owners have enjoyed. The ROM board isn't cheap, but it includes all the address decoding so that future ROMs can be added at minimum cost. It also has slightly more features than the other two versions. It leaves all of the user RAM free for source text and object code – over 38K! Most important for me, being in ROM means I can trust it totally. When using any RAM-based assembler/monitor, there's always the nagging doubt that a bug in your program may have corrupted the monitor code somewhere: a timebomb ready to crash the system when it's least convenient. Speaking of bugs, Maxam is my workhorse and I haven't found one yet. It's a real pleasure to use.

Maxam can be used in several ways. First, all the source code be be included in REM statements in a Basic program, allowing you to mix the two as does BBC Basic. (Though the assembler is built-in to BBC Basic, so the source doesn't have to be 'hidden' in REMs).

At run-time the command ASSEM generates and stores the object code which can then be called from the BASIC program. The limitation here is that such hybrid programs can only be run on Amstrads that have Maxam installed, so it's no use for commercial software.

But most commercial products are pure machine code anyway, and Maxam can be used as a normal assembler and monitor too. Its text editor is the best I've seen – it works just like the better word processors, and if it was possible to embed printer control codes in the text file, Arnor could probably sell it as one. You can use it to write *any* text, not just assembler: and one clever command lets you place line numbers at the start of each line. So you have an 'ideas processor' for Basic programs.

Arnor is producing the sort of software that should be encouraged. If you're serious about your programming, hang the expense: buy one now.

THE CODE MACHINE

THE CODE MACHINE

Cassette £19.95
Picturesque 01 777 0572

The final offering comes from Picturesque, another company noted for past assembler offerings on the Spectrum. The Code Machine consists of two independent programs on either side of a cassette, AMMAS and AMMON. AMMON is about 7K, AMMAS a bit longer, and the two prgrams may be loaded together or independently. Each is fully relocatable and is installed as a Resident System Extension, the commands being MON and ASS (unfortunate abbreviation). Disk users are given the option to make a working copy of the programs on disk. Good.

The Code Machine has a lot of options not available on any of the other programs. It is also the most compatible with the CPC464, in the sense that things work in the same way as when editing Basic. For example, when editing the source text you can use the COPY cursor – none of the other assemblers allows this – and line numbers are obligatory, but act like Basic numbers.

Picturesque gets a black mark for the format of hexadecimal numbers, which must begin with a number and end in H.

One welcome addition is a Verify command for tape or disk files, the code for which is present in the firmware but inexplicably absent from Locomotive Basic.

The monitor has all the usual features plus some rather nice goodies. For example, TRACE (step mode) has a remarkably full screen display which includes normal and alternate registers and flags, current upper ROM number and both ROM states, the areas indexed by BC, DE and HL (shame about IX and IY, though) the next nine bytes of the program, and five levels of stack contents. Picturesque also claims that Trace can operate with any value in any register, despite Amstrad's insistence that the machine will crash if BC or the alternate carry are tampered with.

The manual is as big as that for ZEN, 68 pages, but it's all solid instruction and examples – lots of examples, a trend to be encourages. If I wasn't in love with Maxam, The Code Machine could well turn by head.

Power supply design

Selecting power supply components on a 'suck it and see' basis is not recommended – an under-rated capacitor could well explode. Nigel Eames' program takes the guesswork out of psu design.

The choice of power supply components is not a simple and straightforward matter; potential disaster awaits anyone using electrolytic capacitors for smoothing in power supplies if the ripple rating of the capacitor is less than that required for the job. There is a serious risk that the smoothing capacitor will explode if it is underrated. (The ripple rating of the capacitor is the maximum allowable rms current that the capacitor can handle.) The rms ripple current is greater than the load current due to the complex nature of the current waveform, and so it is not sufficient to choose a capacitor with a ripple rating equal to the load current. Selecting smoothing capacitors on a "suck it and see" basis can be dangerous on the one hand if the component is under-rated, and expensive and unnecessarily bulky on the other if it is over rated.

Similarly, haphazard selection of the transformer and diode ratings can result in failure though usually less spectacularly. This article describes the factors that determine ripple current in the capacitor and the effect it has on the other components. A program is given to calculate the ripple current, ripple voltage, peak repetitive diode current, average diode current, peak dc output voltage and the required transformer VA rating so that suitable components can be selected for a particular application.

Very often the requirement is for a circuit similar to that in **Figure 1.** The load current is usually a variable quantity but for the purposes of calculating the ripple current the "worst case" is assumed, that is maximum load current. If the load is a regulator chip, as is very often the case, the load current is independent of the capacitor voltage, assuming of course that the minimum required potential across the regulator chip is maintained. If the load is simply a resistor then the load current will only remain approximately constant as long as the capacitor is not allowed to discharge significantly.

The voltage across the smoothing capacitor in a typical power supply is a combination of part of a cosine curve and a linear negative-going ramp which corresponds to the charging cycle and discharging cycles respectively. The cosine portion of the curve results from the output voltage from the diode bridge exceeding the capacitor voltage, causing current to flow into the capacitor charging it up to the peak supply voltage. The linear portion is the capacitor discharge cycle – as the bridge output voltage is lower than the capacitor voltage during this time. The diodes are reverse biased and are therefore non-conducting and it is the capacitor which supplies the load current, but as it does so the voltage across it decreases.

Vcap = Vpk–I*t/C for constant current loads and
Vcap = Vpk*exp(–t/CR) for resistive loads

In practice there is only a small difference in the required ratings for the components in the two cases, the constant current case being the more exacting of the two and is considered here. The current waveform across the capacitor consists of two distinct sections. The positive part of the waveform is the charging current and the negative part is the current supplied to the load, ie the discharging current. The charging portion of the waveform is part of a sinusoid but phase shifted with respect to the sinusoidal part of the voltage waveform by +90. The discharging portion is linear and constant as already stated. Clearly the charge supplied to the capacitor during the charging time must equal the charge drawn from the capacitor during the discharging time. If this were not so the average potential on the capacitor would not remain constant. Given a fixed load current, capacitor and transformer voltage it should be possible to make the necessary calculations of the ripple voltage and current. The problem looks deceptively simple until you realise that the relative charge and discharge times are not known and the solution to the equation defining the times is not readily solvable. The equation is:

Vpk–I*t/C = –Vpk*COSwt (1)

Equation (1) defines the point in time at which the linear part of the voltage waveform crosses the cosine curve. Solving it analytically is rather difficult (to say the least!) but it can be solved to any required degree of accuracy using iterative methods if a computer is used to do all the hard work. Having found the time at which these two curves intersect it can be substituted into the relevant formulae to yield the required results.

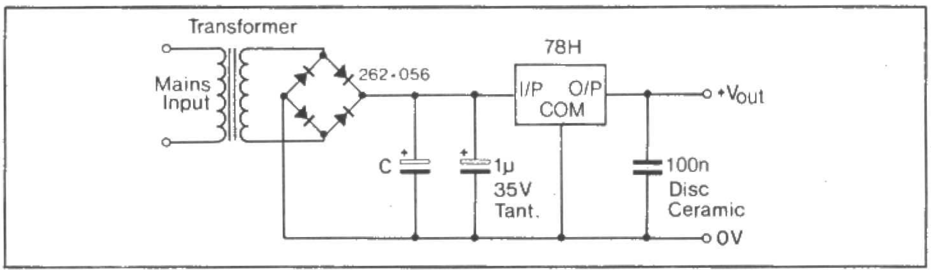


Figure 1. Typical circuit.

Choice of capacitance

A rule of thumb guide to the choice of capacitance value is that:

C*Vave.dc./Iload = 5*T
or
C*R = d*T

ie the time constant of the capacitance times the equivalent load resistance should be about 2.5 times the supply period, ie 50mS for a 50Hz supply. The actual value is inevitably a compromise.

Increasing the time constant reduces the ripple voltage, which is usually desirable, but it has the disadvantage that it increases the ripple current. The reason for this is that increasing the time constant reduces the charging time, consequently the peak charging current is higher and this increases the rms value of the ripple current. It should be remembered at this point that most electrolytic capacitors have a very wide tolerance band, –20% +50% being typical which can lead to wide variations in ripple voltage and ripple current for identical circuits.

The complicated current waveform imposes restrictions on the diodes and transformer used in the supply. The average diode current must equal the load current but what is not so obvious is that the peak diode current can be well over an order of magnitude higher than the load current. The peak diode current will be equal to the peak charging current of the capacitor plus the load current because the diodes not only pass current to charge the capacitor but during this charging time provide the load current as well. (The diode peak inverse voltage rating should be equal to twice the peak transformer output voltage.)

The transformer VA rating is also affected by the current waveform. The VA rating is a product of the rms output voltage multiplied by the rms output current. The rms current from the transformer can be many times the load current and must be taken into account if the transformer is not to be overloaded.

The program

The main program is contained in lines 40 to 160. The first procedure called at line 40 displays a two option menu. This allows the choice between a computer generated solution to the selection of the power supply components or, a check of an existing design.

If the first option is chosen the program requests the required output voltage from the unregulated supply and the load current. The calculations are then carried out assuming that the transformer regulation is 10% (line 1400). This is a typical value though it can be as high as 25% for some transformers. The CR time constant (ie the smoothing capacitance multiplied by the equivalent load resistance) is taken as 50mS (line 1400).

If the second option is chosen the program requests the transformer output voltage and regulation, the load current and the smoothing capacitor value. Each quantity entered is checked for reasonable values and a re-enter prompt issued if the data entered is inconsistent. For example if the capacitor value entered is so small that the potential across it would drop to zero in less than 5mS a new value is requested.

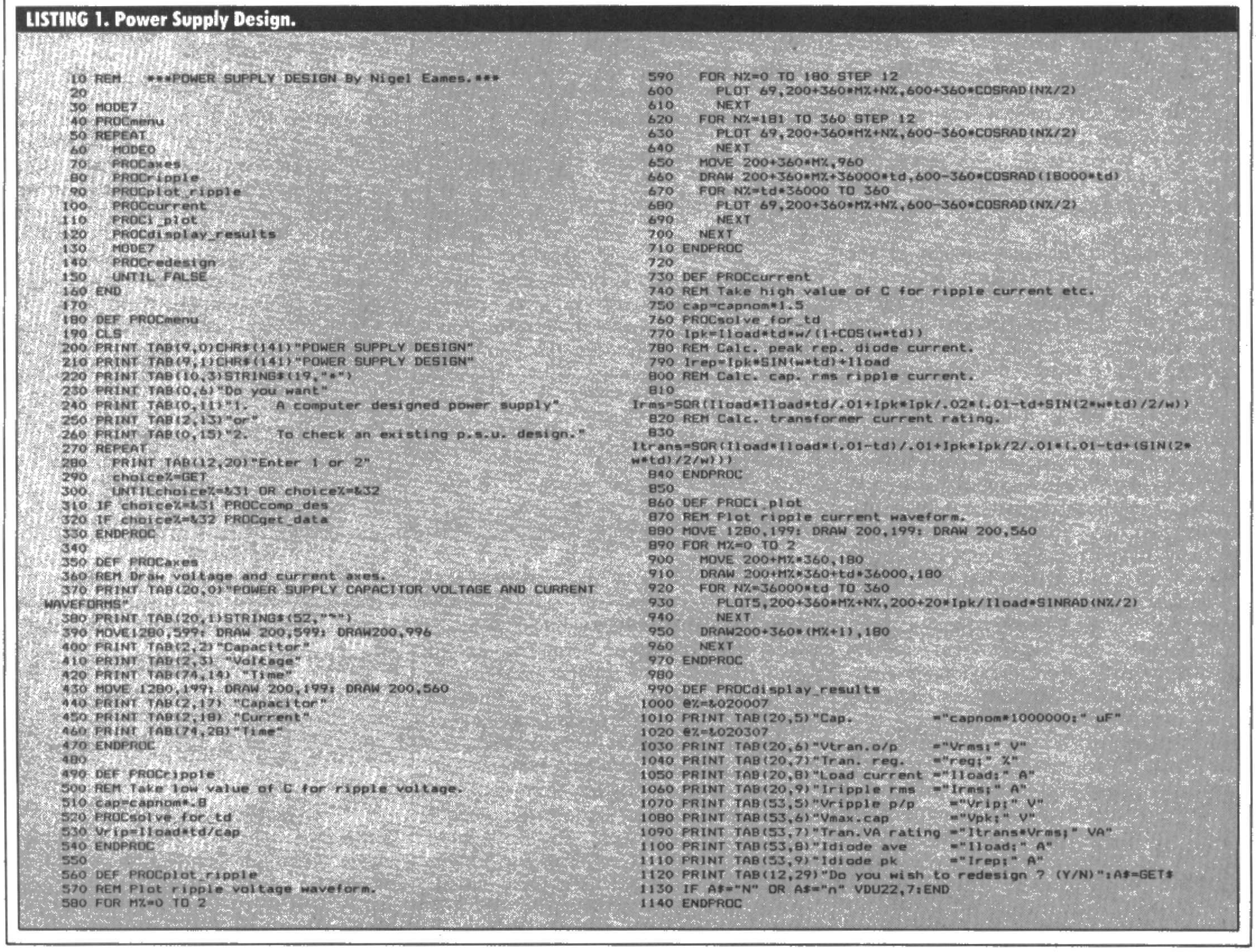
The calculations for the supply ripple voltage are carried out assuming the capacitor tolerance is 20% (lines 640-810). This is the worst case as far as ripple voltage is concerned.

Calculations for the ripple current are performed assuming that the capacitor tolerance is +50% (lines 740-750). This is the worst case condition for the ripple current as the rms ripple current increases with increased capacitance.

The program generates a display of the supply ripple voltage and capacitor ripple current waveforms taking the peak capacitor voltage and the load current as references respectively. The component current and voltage ratings are then displayed along with the option of a re-design facility if required. The re-design procedure allows any or all of the four main parameters to be changed.

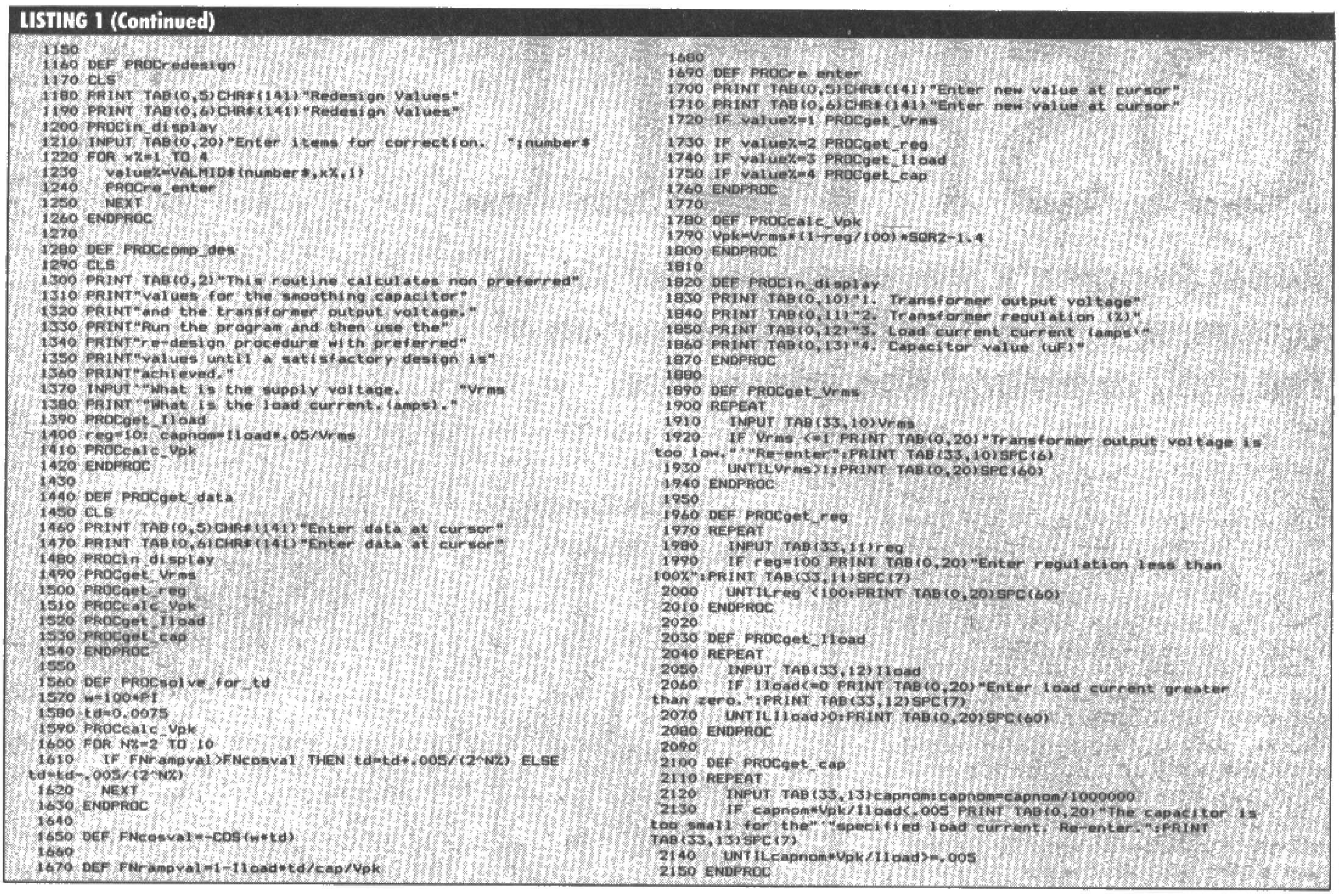
All the procedures in the program are self explanatory with the exception of PROCsolve–for–td, (line 1560). This procedure is used for finding the relative charge and discharge times for the capacitor. It uses two functions defined immediately after the procedure itself at lines 1650 and 1670, each of which is one side of the equation (1) defined earlier. The procedure owes its inspiration to successive approximation analogue to digital convertors and gives a solution for the discharge time correct to within 0.1%.

LISTING 1. Power Supply Design.



```
   10 REM   ***POWER SUPPLY DESIGN By Nigel Eames.***
   20
   30 MODE7
   40 PROCmenu
   50 REPEAT
   60   MODE0
   70   PROCaxes
   80   PROCripple
   90   PROCplot_ripple
  100   PROCcurrent
  110   PROCi_plot
  120   PROCdisplay_results
  130   MODE7
  140   PROCredesign
  150   UNTIL FALSE
  160 END
  170
  180 DEF PROCmenu
  190 CLS
  200 PRINT TAB(9,0)CHR$(141)"POWER SUPPLY DESIGN"
  210 PRINT TAB(9,1)CHR$(141)"POWER SUPPLY DESIGN"
  220 PRINT TAB(10,3)STRING$(19,"*")
  230 PRINT TAB(0,6)"Do you want"
  240 PRINT TAB(0,11)"1.   A computer designed power supply"
  250 PRINT TAB(2,13)"or"
  260 PRINT TAB(0,15)"2.   To check an existing p.s.u. design."
  270 REPEAT
  280   PRINT TAB(12,20)"Enter 1 or 2"
  290   choice%=GET
  300   UNTILchoice%=&31 OR choice%=&32
  310 IF choice%=&31 PROCcomp_des
  320 IF choice%=&32 PROCget_data
  330 ENDPROC
  340
  350 DEF PROCaxes
  360 REM Draw voltage and current axes.
  370 PRINT TAB(20,0)"POWER SUPPLY CAPACITOR VOLTAGE AND CURRENT
WAVEFORMS"
  380 PRINT TAB(20,1)STRING$(52,"~")
  390 MOVE1280,599: DRAW 200,599: DRAW200,996
  400 PRINT TAB(2,2)"Capacitor"
  410 PRINT TAB(2,3) "Voltage"
  420 PRINT TAB(74,14) "Time"
  430 MOVE 1280,199: DRAW 200,199: DRAW 200,560
  440 PRINT TAB(2,17) "Capacitor"
  450 PRINT TAB(2,18) "Current"
  460 PRINT TAB(74,28)"Time"
  470 ENDPROC
  480
  490 DEF PROCripple
  500 REM Take low value of C for ripple voltage.
  510 cap=capnom*.8
  520 PROCsolve_for_td
  530 Vrip=Iload*td/cap
  540 ENDPROC
  550
  560 DEF PROCplot_ripple
  570 REM Plot ripple voltage waveform.
  580 FOR M%=0 TO 2
  590   FOR N%=0 TO 180 STEP 12
  600     PLOT 69,200+360*M%+N%,600+360*COSRAD(N%/2)
  610     NEXT
  620   FOR N%=181 TO 360 STEP 12
  630     PLOT 69,200+360*M%+N%,600-360*COSRAD(N%/2)
  640     NEXT
  650   MOVE 200+360*M%,960
  660   DRAW 200+360*M%+36000*td,600-360*COSRAD(18000*td)
  670   FOR N%=td*36000 TO 360
  680     PLOT 69,200+360*M%+N%,600-360*COSRAD(N%/2)
  690     NEXT
  700   NEXT
  710 ENDPROC
  720
  730 DEF PROCcurrent
  740 REM Take high value of C for ripple current etc.
  750 cap=capnom*1.5
  760 PROCsolve_for_td
  770 Ipk=Iload*td*w/(1+COS(w*td))
  780 REM Calc. peak rep. diode current.
  790 Irep=Ipk*SIN(w*td)+Iload
  800 REM Calc. cap. rms ripple current.
  810
Irms=SQR(Iload*Iload*td/.01+Ipk*Ipk/.02*(.01-td+SIN(2*w*td)/2/w))
  820 REM Calc. transformer current rating.
  830
Itrans=SQR(Iload*Iload*(.01-td)/.01+Ipk*Ipk/2/.01*(.01-td+(SIN(2*
w*td)/2/w)))
  840 ENDPROC
  850
  860 DEF PROCi_plot
  870 REM Plot ripple current waveform.
  880 MOVE 1280,199: DRAW 200,199: DRAW 200,560
  890 FOR M%=0 TO 2
  900   MOVE 200+M%*360,180
  910   DRAW 200+M%*360+td*36000,180
  920   FOR N%=36000*td TO 360
  930     PLOT5,200+360*M%+N%,200+20*Ipk/Iload*SINRAD(N%/2)
  940     NEXT
  950   DRAW200+360*(M%+1),180
  960   NEXT
  970 ENDPROC
  980
  990 DEF PROCdisplay_results
 1000 @%=&020007
 1010 PRINT TAB(20,5)"Cap.           ="capnom*1000000;" uF"
 1020 @%=&020307
 1030 PRINT TAB(20,6)"Vtran.o/p    ="Vrms;" V"
 1040 PRINT TAB(20,7)"Tran. reg.   ="req;" %"
 1050 PRINT TAB(20,8)"Load current ="Iload;" A"
 1060 PRINT TAB(20,9)"Iripple rms  ="Irms;" A"
 1070 PRINT TAB(53,5)"Vripple p/p     ="Vrip;" V"
 1080 PRINT TAB(53,6)"Vmax.cap        ="Vpk;" V"
 1090 PRINT TAB(53,7)"Tran.VA rating ="Itrans*Vrms;" VA"
 1100 PRINT TAB(53,8)"Idiode ave      ="Iload;" A"
 1110 PRINT TAB(53,9)"Idiode pk       ="Irep;" A"
 1120 PRINT TAB(12,29)"Do you wish to redesign ? (Y/N)":A$=GET$
 1130 IF A$="N" OR A$="n" VDU22,7:END
 1140 ENDPROC
```

LISTING 1 (Continued)



```
 1150
 1160 DEF PROCredesign
 1170 CLS
 1180 PRINT TAB(0,5)CHR$(141)"Redesign Values"
 1190 PRINT TAB(0,6)CHR$(141)"Redesign Values"
 1200 PROCin_display
 1210 INPUT TAB(0,20)"Enter items for correction.  ";number$
 1220 FOR x%=1 TO 4
 1230   value%=VALMID$(number$,x%,1)
 1240   PROCre_enter
 1250   NEXT
 1260 ENDPROC
 1270
 1280 DEF PROCcomp_des
 1290 CLS
 1300 PRINT TAB(0,2)"This routine calculates non preferred"
 1310 PRINT"values for the smoothing capacitor"
 1320 PRINT"and the transformer output voltage."
 1330 PRINT"Run the program and then use the"
 1340 PRINT"re-design procedure with preferred"
 1350 PRINT"values until a satisfactory design is"
 1360 PRINT"achieved."
 1370 INPUT'"What is the supply voltage.      "Vrms
 1380 PRINT'"What is the load current.(amps)."
 1390 PROCget_Iload
 1400 reg=10: capnom=Iload*.05/Vrms
 1410 PROCcalc_Vpk
 1420 ENDPROC
 1430
 1440 DEF PROCget_data
 1450 CLS
 1460 PRINT TAB(0,5)CHR$(141)"Enter data at cursor"
 1470 PRINT TAB(0,6)CHR$(141)"Enter data at cursor"
 1480 PROCin_display
 1490 PROCget_Vrms
 1500 PROCget_reg
 1510 PROCcalc_Vpk
 1520 PROCget_Iload
 1530 PROCget_cap
 1540 ENDPROC
 1550
 1560 DEF PROCsolve_for_td
 1570 w=100*PI
 1580 td=0.0075
 1590 PROCcalc_Vpk
 1600 FOR N%=2 TO 10
 1610   IF FNrampval>FNcosval THEN td=td+.005/(2^N%) ELSE
td=td-.005/(2^N%)
 1620   NEXT
 1630 ENDPROC
 1640
 1650 DEF FNcosval=-COS(w*td)
 1660
 1670 DEF FNrampval=1-Iload*td/cap/Vpk
 1680
 1690 DEF PROCre enter
 1700 PRINT TAB(0,5)CHR$(141)"Enter new value at cursor"
 1710 PRINT TAB(0,6)CHR$(141)"Enter new value at cursor"
 1720 IF value%=1 PROCget_Vrms
 1730 IF value%=2 PROCget_reg
 1740 IF value%=3 PROCget_Iload
 1750 IF value%=4 PROCget_cap
 1760 ENDPROC
 1770
 1780 DEF PROCcalc_Vpk
 1790 Vpk=Vrms*(1-reg/100)*SQR2-1.4
 1800 ENDPROC
 1810
 1820 DEF PROCin_display
 1830 PRINT TAB(0,10)"1. Transformer output voltage"
 1840 PRINT TAB(0,11)"2. Transformer regulation (%)"
 1850 PRINT TAB(0,12)"3. Load current current (amps)"
 1860 PRINT TAB(0,13)"4. Capacitor value (uF)"
 1870 ENDPROC
 1880
 1890 DEF PROCget_Vrms
 1900 REPEAT
 1910   INPUT TAB(33,10)Vrms
 1920   IF Vrms <=1 PRINT TAB(0,20)"Transformer output voltage is
too low."'"Re-enter":PRINT TAB(33,10)SPC(6)
 1930   UNTILVrms>1:PRINT TAB(0,20)SPC(60)
 1940 ENDPROC
 1950
 1960 DEF PROCget_reg
 1970 REPEAT
 1980   INPUT TAB(33,11)reg
 1990   IF reg=100 PRINT TAB(0,20)"Enter regulation less than
100%":PRINT TAB(33,11)SPC(7)
 2000   UNTILreg <100:PRINT TAB(0,20)SPC(60)
 2010 ENDPROC
 2020
 2030 DEF PROCget_Iload
 2040 REPEAT
 2050   INPUT TAB(33,12)Iload
 2060   IF Iload<=0 PRINT TAB(0,20)"Enter load current greater
than zero.":PRINT TAB(33,12)SPC(7)
 2070   UNTILIload>0:PRINT TAB(0,20)SPC(60)
 2080 ENDPROC
 2090
 2100 DEF PROCget_cap
 2110 REPEAT
 2120   INPUT TAB(33,13)capnom:capnom=capnom/1000000
 2130   IF capnom*Vpk/Iload<.005 PRINT TAB(0,20)"The capacitor is
too small for the"'"specified load current. Re-enter.":PRINT
TAB(33,13)SPC(7)
 2140   UNTILcapnom*Vpk/Iload>=.005
 2150 ENDPROC
```

QSHELL

Adam Denning's magnum opus for the QL continues with a detailed examination of the shell command line interpreter.

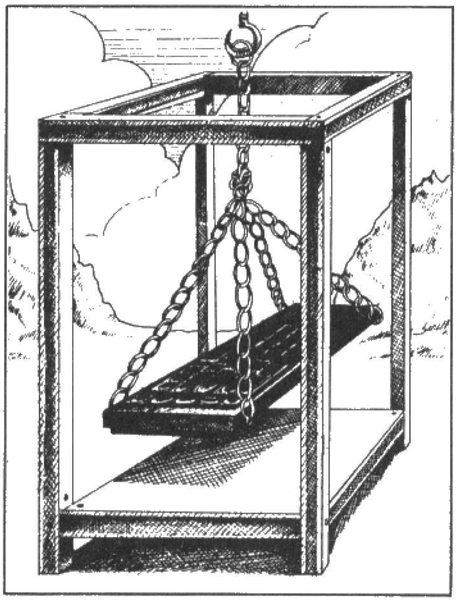
Last month's article introduced the SHELL initialisation routines, and briefly mentioned how the command line interpreter (CLI) itself worked. Now is the time for a more detailed examination of the CLI in this and the remaining codes.

A word of warning: QSHELL is still very much under development, and there are a couple of things about the way the CLI is coded which I intend to alter. First, it has a bug which manifests itself if you type in extremely long command lines. Second, I decided early on to make it convert all command words to a special format, in which each word is a string of bytes terminated by a zero byte. This is not as sensible as I first thought, but it'll take a long time to alter all the routines. If either of these is fixed by the time the final article is published *(and they better be, Ed)* the requisite corrections will be given.

So let's take a look at the code involved in the CLI. The first stage is to collect the command line from channel 0, which we do at **COMMAND.** This gets the channel ID from the **CHAN_0** variable and uses SD_CURE to enable the cursor.

Although the line will be collected with the IO_FLINE trap, which enables the cursor itself, it is as well to enable the cursor now so that there will be no problems with keyboard queue switching.

The routine to grab the command line is **GET_COMLN,** which prints a '>' cursor and calls IO_FLINE. The buffer used to collect the command line is at the very beginning of the job's data space. If the line consists only of a line feed character, the routine calls IO_FLINE again. If the line entered is so long that IO_FLINE returns ERR_ BF, a message saying that the line is too long is printed, and IO_EDLIN is called to re-present the line. Eventually, GET_COMLN will return with a suitable command line pointed to by A1. Any groups of two or more adjacent spaces are then converted to single spaces by calling **REMOVE,** which replaces any given two-character pair with a given single character. If this results in there being only one character left, and this character turns out to be a space, then the line is effectively blank so the routine branches back to the start.



THE STORY SO FAR

QSHELL is a major software project designed to provide QDOS with an MSDOS type front-end. The extensive program provides wildcard facilities for file copying, deletion and renaming. Another aspect of QShell is the provision of easy access to QDOS features such as multi-tasking, job control and device independent I/O.

The program offers an extensive example of the use of 68000 assembly language and many sections of the program can be used as useful additions to the QL's resident firmware.

If you have missed either of the first two installments of the series, copies of the relevant issues of *E&CM* (June and July) are available from our back issues service, Tel: 0858 34567.

The line is then checked to see if it has a trailing space, and this is removed if present. The line last entered is then stored in one of the command line stores by converting the value **COM_LEVL** into a pointer to the requisite command line store. **STRING_MV,** which moves a string from (A2) to A4), is called to actually store the command line.

The command line is then passed to SPLIT_CLN, which finds the command word and separates it from any command tail. The command word is stored directly after the original command line in the data space, and we use the subroutine NEW_HEAP to calculate the requisite pointer to it. The command word is converted to upper case as it is moved, and the result is stored as a 'SHELL string', which is a sequence of bytes terminated by a zero byte. Well it seemed like a good idea at the time! If there is a command tail after the command word, this is moved as a normal QDOS string to the data space, being stored immediately after the command word. If there is no command tail, A1 is left with 0 in it.

A command line of:

run myprog 'parameter string' input output

will be converted into a SHELL string:

RUN

pointed to by A0, and a QDOS string command tail:

myprog 'parameter string' input output

pointed to by A1. These addresses are passed to **DO_COMND** for further processing. DO_COMND scans the table of QSHELL commands until it matches the command word and a word in the command tail, or until it reaches the end of the table. Each entry in the table is stored as follows:

Characters of name
Byte of 0
[align to word boundary if necessary]
Word offset from end of table to start of routine

If the command word matches an entry, the routine's offset is put into D0 and used as an offset from A2 so that 0(A2,D0.L) points to the start of the command word's routine. If the command word does not match an entry, the routine branches to **NOTCWORD,** which attempts to treat it as a filename.

NOTCWORD first tries to open the file in its own right, but if this fails it appends the default device name and tries again. If this fails as well, the command word is invalid, so a message is printed, D0 is set to 1 (to indicate an error), and the routine returns via **COMEBACK.**

If the file can be opened, its header is read in and checked to ensure that it has file type 1 (executable file). If not, an error is reported. Otherwise, the file's length and default data space length are read and used in a call to **MT_CJOB,** which creates a job. If there is no problem with this, the file is loaded into the designated area of memory with **FS_LOAD** and the file is closed. If all is still going well, the command tail will be passed to **OPEN_JCHN** which opens any channels the job requires and passes it any parameter string. This is done in a way which is compatible with any 'standard' job which is executed by EX if you have Tony Tebby's QL Toolkit. Finally, the job is activated with a timeout of –1 (equivalent to EXEC_W or EW).

All routines and programs will eventually return to the main program at COMEBACK. If D0 is not zero and the SHELL system variable **ERR_STAT** is non-zero (ie. if the error level is set), then a message is printed so that the line may be repeated, edited or ignored. This is similar to the MS-DOS message Abort, Retry, Ignore? but is a little more useful. Pressing 'R' sends the complete line back to the CLI so that the operation can be repeated, 'E' sends the command line to IO_EDLIN so that it may be re-edited, and 'I' or Enter ignores the error condition and returns to the normal prompt. Notice how the PEA ('push effective address') instruction is used to push the address of part of the command line routine onto the stack ready for an RTS.

Two relevant command routines are shown here as well: the routine to load a job without executing it, and the routine to load a job and execute it with zero timeout, equivalent to EXEC or EX. Register D7 is used in both these routines, as it was in NOTCWORD, to indicate whether the job should be activated yet and with what timeout (jobs are always activated with an initial priority of 32, as it seems as good a value as any).

The utility routines **REMOVE, GET_ARGN** and **OPEN_JCHN** are fairly complex. REMOVE is entered with the two character values in D3 and D4, and another character value in D5. It scans the QDOS string pointed to by A1 looking for adjacent occurences of D3 and D4, and replaces the pair by the character in D5, reducing the length of the string by one each time a pair is replaced. It continually loops until every pair is removed from the string, and so is ideal for removing groups of spaces from command lines.

'QSHELL error messages are similar to the MS-DOS Abort, Retry, Ignore? but a little more useful.

GET_ARGN separates the nth argument from a command string. D1 holds 'n' and D2 holds the separator, which is normally a space. The original string, pointed to by A1, is unchanged, and the new string containing one argument is pointed to by A0.

OPEN_JCHN is entered with a lot of information on the stack relevant to a job, such as its base address and job ID. It first searches the command tail for an option string, which is always delimited by single quotes and must be the first item in the job's parameter list:

RUN myjob 'option string' input_file output_file

If the string is found, it is moved complete to the job's stack in accordance with Sinclair standard format. The routine then scans the command string for the second occurence of a space-delimited string. If found, this is used as the job's output channel. It is opened as a new file by OPEN_THIS, which will append the default device name if appropriate. The resultant channel ID is stack, and D6 is incremented. Finally, the routine looks for an input channel filename. If this is found, it is opened and the channel ID is stacked. Then D6.W is stacked, so that when the job is activated, it will find this information on its stack:

Word: number of channels opened for this job
Long: channel ID of input channel
Long: channel ID of output channel
Word: Length of option string
Bytes: characters of option string

Of course, there may have been no parameter list, in which case the values stacked by QDOS (two words of zero) tell the job that it has no channels and no option string.

LISTING 1. QSHELL command line interpreter procedures.

```
SPLIT_CLN MOVEQ     #3,D0
          MOVE.L    A1,-(A7)
          MOVE.W    D1,D2
          BSR       NEW_HEAP
          MOVEA.L   A1,A0
          MOVEA.L   (A7)+,A1
          MOVEQ     #0,D0
          MOVEQ     #0,D1
          MOVEA.L   A0,A2
          ADDQ.L    #2,A1
          ADDQ.W    #1,D2
STOF1st   MOVE.B    (A1)+,D1
          SUBQ.W    #1,D2
          CMPI.B    #' ',D1
          BEQ.S     STOF1st
          BSR.S     TOUPPER
          MOVE.B    D1,(A0)+
          ADDQ.W    #1,D0
          CMP.W     D0,D2
          BEQ.S     ENDOF1st
DURING1st MOVE.B    (A1)+,D1
          CMPI.B    #' ',D1
          BEQ.S     ENDOF1st
          BSR.S     TOUPPER
          MOVE.B    D1,(A0)+
          ADDQ.W    #1,D0
          CMP.W     D0,D2
          BNE.S     DURING1st
ENDOF1st  CLR.B     (A0)
          MOVEA.L   A2,A0
          LEA.L     D_SPACE,A2
          MOVEA.L   A2,A3
          SUBQ.W    #1,D2
          SUB.W     D0,D2
          BPL.S     COM_TAIL
          SUBA.L    A1,A1
          RTS
COM_TAIL  MOVE.W    D2,(A2)+
          SUBQ.W    #1,D2

TAIL_MOVE MOVE.B    (A1)+,(A2)+
          DBRA      D2,TAIL_MOVE
          MOVEA.L   A3,A1
          RTS
TOUPPER   CMPI.B    #'a',D1
          BLT.S     NOTCAPTL
          CMPI.B    #'z',D1
          BGT.S     NOTCAPTL
          SUBI.B    #32,D1
NOTCAPTL  RTS
EOFCWORD  ADDQ.L    #3,A1
          MOVE.L    A1,D0
          ANDI.L    #-2,D0
          MOVEA.L   D0,A1
          BRA.S     FINDCWD
GOTCWORD  TST.B     (A0)
          BEQ.S     REALLY_G
          SUBQ.L    #1,A1
          BRA.S     WRONG_WD
REALLY_G  MOVEQ     #0,D0
          MOVE.L    A1,D2
          ADDQ.L    #1,D2
          ANDI.L    #-2,D2
          MOVE.L    D2,A1
          MOVE.W    (A1),D0
          MOVEA.L   (A7)+,A1
          MOVEA.L   A3,A0
          JSR       2(A2,D0.L)
*
*
COMEBACK  TST.L     D0
          BEQ       CMD_END
          LEA.L     ERR_STAT,A1
          TST.W     (A1)
          BEQ       CMD_END
          LEA.L     RPT_MES,A1
          MOVEA.L   CHAN_0,A0
          VECTOR    UT_MTEXT,2,CALL
          MOVEQ     #-1,D3

          QDOS      SD_CURE,3
C_LETTER
          QDOS      IO_FBYTE,3
          BSR       TOUPPER
          CMPI.B    #'R',D1
          BEQ.S     AGN_CMDR
          CMPI.B    #'E',D1
          BEQ.S     AGN_EDCMD
          CMPI.B    #linefeed,D1
          BEQ.S     IGNORE_L
          CMPI.B    #'I',D1
          BNE.S     C_LETTER
          MOVEQ     #linefeed,D1
IGNORE_L
DO_COMND  MOVE.L    A1,-(A7)
          LEA.L     CWORDTAB,A1
          LEA.L     ENDCWTAB-2,A2
FINDCWD   MOVEA.L   A0,A3
          CMPA.L    A2,A1
          BHI.S     WD_WRNG
CHKCWORD  MOVE.B    (A1)+,D0
          BEQ.S     GOTCWORD
          CMP.B     (A0)+,D0
          BEQ.S     CHKCWORD
WRONG_WD  MOVEA.L   A3,A0
NOT_EOFC  TST.B     (A1)+
          BEQ.S     EOFCWORD
          CMPA.L    A1,A2
          BCC.S     NOT_EOFC
WD_WRNG   MOVEQ     #-1,D7
          BRA       NOTCWORD
          QDOS      IO_SBYTE,3
          BRA.S     CMD_END
AGN_CMDR  MOVEQ     #linefeed,D1
          QDOS      IO_SBYTE,3
          MOVEQ     #0,D1
          MOVE.W    COM_LEVL,D1
          SUBQ.W    #1,D1

          LSL.L     #2,D1
          LEA.L     COM_0,A2
          MOVEA.L   0(A2,D1.L),A1
          MOVE.L    A1,-(A7)
          VECTOR    UT_MTEXT,2,CALL
          MOVEQ     #linefeed,D1

          MOVEQ     #-1,D3
          QDOS      IO_SBYTE,3
          MOVEA.L   (A7)+,A1
          MOVE.W    (A1),D1
          BRA       REPT_CMD
AGN_EDCMD MOVEQ     #linefeed,D1
          QDOS      IO_SBYTE,3
          MOVEQ     #0,D1
          MOVE.W    COM_LEVL,D1
LAST_ED   SUBQ.W    #1,D1
          LSL.L     #2,D1
          LEA.L     COM_0,A2
          MOVEA.L   0(A2,D1.L),A1
          MOVE.W    (A1),D1
          BEQ       LAST_ERR1
          MOVEA.L   A1,A2
          LEA.L     D_SPACE,A4
          BSR       STRING_MV
          PEA.L     STORE_ED
          MOVEM.L   A4/D1,-(A7)
          BRA       RE_EDIT
CMD_END   RTS

RPT_MES   DC.W      70
          DC.B      'The last command
          returned with an error: '
          DC.B      '(R)epeat,
          (E)dit or (I)gnore?'

NOTCWORD  MOVE.L    A0,-(A7)
          BSR       NEW_PTR
          MOVEQ     #-1,D0
```

LISTING 1 (Continued)

```
          MOVE.L    (A7),A1
          MOVE.L    A0,(A7)
          ADDQ.L    #2,A0
MOVECMD   ADDQ.L    #1,D0
          MOVE.B    (A1)+,(A0)+
          BNE.S     MOVECMD
          MOVE.L    (A7),A0
          MOVE.W    D0,(A0)
          MOVEQ     #-1,D1
          MOVEQ     #OPEN_INS,D3
          QDOS      IO_OPEN,2
          TST.L     D0
          BEQ.S     FILECMD
          MOVE.L    (A7),A1
          MOVEQ     #3,D0
          BSR       NEW_HEAP
          MOVEA.L   A1,A4
          MOVEA.L   (A7),A3
          BSR       OPEN_DEF
          BEQ.S     FILECMD
          MOVEM.L   (A7)+,A1/A2
          MOVEA.L   CHAN_0,A0
          VECTOR    UT_MTEXT,2,CALL
          LEA.L     COM_MES,A1
          VECTOR    UT_MTEXT,2,CALL
          MOVEQ     #1,D0
          BRA       COMEBACK
FILECMD   MOVEA.L   (A7),A1
          MOVEQ     #3,D0
          BSR       NEW_HEAP
          MOVE.L    A1,-(A7)
          MOVEQ     #-1,D3
          MOVEQ     #16,D2
          QDOS      FS_HEADR,3
          MOVEA.L   (A7)+,A1
          TST.L     D0
          BNE.S     CFL_ERR
          CMPI.B    #1,5(A1)
          BEQ.S     CFL_TYP
          QDOS      IO_CLOSE,2
          MOVEM.L   (A7)+,A1/A2
          MOVEA.L   CHAN_0,A0
          VECTOR    UT_MTEXT,2,CALL
          LEA.L     COM_MES1,A1
          VECTOR    UT_MTEXT,2,CALL
          MOVEQ     #1,D0
          BRA       COMEBACK
CFL_TYP   MOVE.L    (A1),D2
          MOVE.L    6(A1),D3
          MOVEQ     #-1,D1
          SUBA.L    A1,A1
          MOVE.L    A0,-(A7)
          QDOS      MT_CJOB,1
          TST.L     D0
          BEQ.S     CFL_COK
          MOVEA.L   (A7)+,A0
CFL_ERR   MOVE.L    D0,D4
          QDOS      IO_CLOSE,2
CFL_ERR1  MOVEM.L   (A7)+,A1/A2
          MOVEA.L   CHAN_0,A0
          VECTOR    UT_MTEXT,2,CALL
          LEA.L     COM_MES2,A1
          VECTOR    UT_MTEXT,2,CALL
          MOVE.L    D4,D0
          VECTOR    UT_ERR,2,CALL
          MOVEQ     #1,D0
          BRA       COMEBACK
CFL_COK   MOVEA.L    A0,A1
          MOVEA.L   (A7),A0
          MOVE.L    A1,(A7)
          MOVE.L    D1,-(A7)
          MOVEQ     #-1,D3
          QDOS      FS_LOAD,3
          MOVE.L    D0,D4
          QDOS      IO_CLOSE,2
          TST.L     D4
          BEQ.S     LD_OK_J
OP_J_ERR  MOVE.L    (A7)+,D1
          MOVEQ     #0,D3
          QDOS      MT_RJOB,1
          ADDQ.L    #4,A7
          BRA.S     CFL_ERR1
LD_OK_J   MOVE.L    12(A7),D0
          BEQ.S     NO_STK_T
          CMPI.B    #2,D7
          BEQ.S     NO_STK_T
          BSR       OPEN_JCHN
```

```
          BNE.S     OP_J_ERR
NO_STK_T  MOVEQ     #0,D0
          MOVE.L    (A7)+,D1
          MOVEM.L   (A7)+,A0-A2
          CMPI.B    #2,D7
          BEQ.S     NOT_YETA
          MOVEQ     #32,D2
          MOVE.L    D7,D3
          QDOS      MT_ACTIV,1
NOT_YETA  BRA       COMEBACK

COM_MES   DC.W      48
          DC.B      ' is not a command and cannot
                  be found as a file',10

COM_MES1  DC.W      27
          DC.B      ' is not an executable file',10,0

COM_MES2  DC.W      18
          DC.B      ' cannot be run as

* Command word table including offsets

CWORDTAB  DC.B      'HELP',0               HELP command
          DC.W      HELP_RTN-ENDCWTAB
          DC.B      'DIR',0                DIR command
          DC.W      DIR_RTN-ENDCWTAB
          DC.B      'TYPE',0               TYPE command
          DC.W      TYPE_RTN-ENDCWTAB
          DC.B      'BASIC',0              BASIC command
          DC.W      QUIT_RTN-ENDCWTAB
          DC.B      'REN',0                RENAME command
          DC.W      REN_RTN-ENDCWTAB
          DC.B      'RENAME',0             RENAME command
          DC.W      REN_RTN-ENDCWTAB
          DC.B      'FORMAT',0             FORMAT command
          DC.W      FMT_RTN-ENDCWTAB
          DC.B      'DEL',0                DELETE command
          DC.W      DEL_RTN-ENDCWTAB
          DC.B      'DELETE',0             DELETE command
          DC.W      DEL_RTN-ENDCWTAB
          DC.B      'WINDOWS',0            WINDOWS command
          DC.W      WNDW_RTN-ENDCWTAB
          DC.B      'CLS',0                CLS command
          DC.W      CLS_RTN-ENDCWTAB
          DC.B      'JOBS',0               JOBS command
          DC.W      JOB_RTN-ENDCWTAB
          DC.B      'USE',0                USE command
          DC.W      USE_RTN-ENDCWTAB
          DC.B      'COPY',0               COPY command
          DC.W      CPY_RTN-ENDCWTAB
          DC.B      'RUN',0                RUN command
          DC.W      RUN_RTN-ENDCWTAB
          DC.B      'EXEC',0               EXEC command
          DC.W      RUN_RTN-ENDCWTAB
          DC.B      'KEY',0                KEY command
          DC.W      KEY_RTN-ENDCWTAB
          DC.B      'NOKEY',0              NOKEY command
          DC.W      NKY_RTN-ENDCWTAB
          DC.B      'KILL',0               KILL command
          DC.W      KIL_RTN-ENDCWTAB
          DC.B      'SUSPEND',0            SUSPEND command
          DC.W      SUS_RTN-ENDCWTAB
          DC.B      'RELEASE',0            RELEASE command
          DC.W      REL_RTN-ENDCWTAB
          DC.B      'BATCH',0              BATCH command
          DC.W      BCH_RTN-ENDCWTAB
          DC.B      'LOAD',0               LOAD command
          DC.W      LOD_RTN-ENDCWTAB
          DC.B      'START',0              START command
          DC.W      STT_RTN-ENDCWTAB
          DC.B      'MEDIUM',0             MEDIUM command
          DC.W      MED_RTN-ENDCWTAB
          DC.B      'MED',0                MEDIUM command
          DC.W      MED_RTN-ENDCWTAB
          DC.B      'FREE',0               FREE command
          DC.W      FRE_RTN-ENDCWTAB
          DC.B      'ERROR',0              ERROR command
          DC.W      ERR_RTN-ENDCWTAB
          DC.B      'LAST',0               LAST command
          DC.W      LST_RTN-ENDCWTAB
          DC.B      'PRIORITY',0           PRIORITY command
          DC.W      PRI_RTN-ENDCWTAP
          DC.B      'MORE',0               MORE command
          DC.W      MRE_RTN-ENDCWTAB
          DC.B      'PRINT',0              PRINT command
          DC.W      PRT_RTN-ENDCWTAB
```

```
          DC.B      'WHERE',0              WHERE command
          DC.W      WHR_RTN-ENDCWTAB
          DC.B      'SIZE',0               SIZE command
          DC.W      SIZ_RTN-ENDCWTAB
          DC.B      'NEW',0                NEW command
          DC.W      NEW_RTN-ENDCWTAB
          DC.B      'READ',0               READ command
          DC.W      RDD_RTN-ENDCWTAB
          DC.B      'WRITE',0              WRITE command
          DC.W      WRT_RTN-ENDCWTAB
          DC.B      'OPEN',0               OPEN command
          DC.W      OPN_RTN-ENDCWTAB
          DC.B      'CLOSE',0              CLOSE command
          DC.W      CLO_RTN-ENDCWTAB
          DC.B      'POINT',0              POINT command
          DC.W      POI_RTN-ENDCWTAB
ENDCWTAB  CNOP      0,2                    end of table
NEW_HEAP  ADD.W     (A1),D0
          ADD.L     A1,D0
          ANDI.L    #-2,D0
          MOVEA.L   D0,A1
          RTS

NEW_PTR   MOVE.B    (A0)+,D0
          BNE.S     NEW_PTR
          ADDQ.L    #1,A0
          MOVE.L    A0,D0
          ANDI.L    #-2,D0
          MOVE.L    D0,A0
          RTS

LOD_RTN   MOVEQ     #2,D7
          BRA.S     LOAD_RUN

RUN_RTN   MOVEQ     #0,D7
LOAD_RUN  MOVE.L    A1,D0
          BNE.S     RUN_CMD
LOAD_ERR  LEA.L     RUN_MES,A1
          MOVEA.L   CHAN_0,A0
          VECTOR    UT_MTEXT,2,CALL
          MOVEQ     #1,D0
          RTS
RUN_CMD   BSR       NEW_CMDS
          TST.B     D7
          BEQ.S     RUN_TAIL
          MOVE.L    A1,D0
          BNE.S     LOAD_ERR
RUN_TAIL  MOVE.L    A1,(A7)
          BRA       NOTCWORD

RUN_MES   DC.W      81
          DC.B      'Please use RUN/LOAD like this:
                    RUN <filename> (<parameters>)'
          DC.B      ' and LOAD <filename>',10,0
NEW_CMDS  LEA.L     D_SPACE,A4
          MOVEA.L   A1,A2
          BSR       STRING_MV
          MOVEA.L   A2,A1
          MOVE.W    (A2),D1
          BRA       SPLIT_CLN

REMOVE    MOVE.L    A1,-(A7)
          MOVE.W    D1,-(A7)
CRS_LOOK  ADDQ.L    #2,A1
          MOVE.W    D1,D0
FIND_CRS  MOVE.B    (A1),D2
          CMP.B     D3,D2
          BEQ.S     GOT_SLSH
          ADDQ.L    #1,A1
          SUBQ.W    #1,D0
          BNE.S     FIND_CRS
NO_LUCK   MOVE.W    (A7)+,D1
          MOVEA.L   (A7)+,A1
          MOVE.W    D1,(A1)
          RTS
GOT_SLSH  ADDQ.L    #1,A1
          SUBQ.W    #1,D0
          BEQ.S     NO_LUCK
          MOVE.B    (A1),D2
          CMP.B     D4,D2
          BNE.S     FIND_CRS
          MOVE.B    D5,-(A1)
          SUBQ.W    #1,(A7)
          SUBQ.W    #2,D0
          BMI.S     N_MVCRS
          MOVEA.L   A1,A2
          ADDQ.L    #2,A2
          ADDQ.L    #1,A1
```

LISTING 1 (Continued)

```
MOVE_CRS   MOVE.B    (A2)+,(A1)+
           DBRA      D0,MOVE_CRS
N_MVCRS    MOVE.W    (A7),D1
           MOVEA.L   2(A7),A1
           MOVE.W    D1,(A1)
           BRA.S     CRS_LOOK

GET_ARGN   MOVEA.L   A1,A3
           MOVEA.L   A1,A2
           MOVEQ     #3,D0
           BSR       NEW_HEAP
           MOVEA.L   A1,A0
           MOVE.W    (A2),D0
           MOVE.W    D0,D5
           SUBQ.W    #1,D0
           MOVEQ     #0,D3
           ADDQ.L    #2,A2
           MOVEA.L   A2,A4
FIND_SEPR  CMP.B     (A2)+,D2
           BEQ.S     GOT_SEPR
           ADDQ.W    #1,D3
SEPR_FIND  DBRA      D0,FIND_SEPR
GOT_SEPR   SUBQ.W    #1,D1
           BEQ.S     GOT_ARG
           TST.W     D0
           BMI.S     NOFND_SEP
           MOVEA.L   A2,A4
           MOVEQ     #0,D3
           BRA.S     SEPR_FIND
NOFND_SEP  MOVEA.L   A3,A1
           RTS
GOT_ARG    MOVE.W    D3,(A1)+
           SUBQ.W    #1,D3
ARG_GOTM   MOVE.B    (A4)+,(A1)+
           DBRA      D3,ARG_GOTM

           MOVEA.L   A3,A1
           MOVE.W    D5,(A1)
           MOVEQ     #0,D0
           RTS
OPEN_JCHN  MOVE.L    16(A7),D1
           BNE.S     PROCESS

           MOVEQ     #0,D4
           RTS
PROCESS    MOVEA.L   8(A7),A0
           SUBA.L    #JOB_AREA-SAVE_USP,A0
           MOVE.L    (A0),-(A7)
           MOVEA.L   D1,A1
           MOVEQ     #2,D1
           MOVEQ     #'''',D2
           BSR       GET_ARGN
           TST.W     D0
           BNE.S     NO_OPTN
           MOVEQ     #3,D0
           ADD.W     (A0),D0
           ANDI.W    #-2,D0
           MOVEA.L   A0,A2
           MOVEA.L   (A7),A4
           SUBA.L    D0,A4
           MOVE.L    A4,(A7)
           BSR       STRING_MV
           MOVEA.L   A1,A4
           MOVEA.L   A1,A3
           MOVEQ     #3,D0
           ADD.W     (A0),D0
           MOVEQ     #0,D6
           SUB.W     D0,(A1)
           BMI       NO_INPCJ
           ADDA.L    D0,A1
           ADDQ.L    #2,A1
           MOVE.W    (A4)+,D0
           BEQ.S     NO_OPTNY
           SUBQ.W    #1,D0

RMV_OPT    MOVE.B    (A1)+,(A4)+
           DBRA      D0,RMV_OPT
NO_OPTNY   MOVEA.L   A3,A1
NO_OPTN    MOVEQ     #0,D6
           TST.W     (A1)
           BEQ.S     NO_INPCJ
           MOVEQ     #2,D1
           MOVEQ     #' ',D2
           BSR       GET_ARGN

           TST.W     D0
           BNE.S     NO_OUTCJ
           MOVEQ     #OPEN_NEW,D3
           MOVE.L    8(A7),D1
           BSR       OPEN_THIS
           TST.L     D0
           BEQ.S     VERY_GOOD

FAIL_OPJ   MOVEA.L   A0,A1
           MOVEA.L   CHAN_0,A0
           MOVE.L    D0,D4
           VECTOR    UT_MTEXT,2,CALL
           MOVEQ     #' ',D1
           MOVEQ     #-1,D3
           QDOS      IO_SBYTE,3
           MOVE.L    D4,D0
           VECTOR    UT_ERR,2,CALL
           ADDQ.L    #4,A7
           LEA.L     THIS_JMES,A0
           MOVE.L    A0,12(A7)
           MOVEQ     #1,D0
           RTS
VERY_GOOD  ADDQ.W    #1,D6
           MOVEA.L   (A7),A1
           MOVE.L    A0,-(A1)
           MOVE.L    A1,(A7)
           MOVEA.L   20(A7),A1
NO_OUTCJ   MOVEQ     #1,D1
           MOVEQ     #' ',D2
           BSR       GET_ARGN
           MOVEQ     #OPEN_INX,D3
           MOVE.L    8(A7),D1
           BSR       OPEN_THIS
           TST.L     D0
           BNE.S     FAIL_OPJ
           ADDQ.W    #1,D6
           MOVEA.L   (A7),A1
           MOVE.L    A0,-(A1)
           MOVE.L    A1,(A7)
NO_INPCJ   MOVEA.L   (A7)+,A1
           MOVE.W    D6,-(A1)

           MOVEA.L   8(A7),A0
           SUBA.L    #JOB_AREA-SAVE_USP,A0
           MOVE.L    A1,(A0)
           MOVEQ     #0,D4
           RTS

THIS_JMES  DC.W      8
           DC.B      'This job'

OPEN_THIS  CMPI.B    #'#',2(A0)
           BNE.S     NOT_HASH
           MOVE.B    3(A0),D1
           CMPI.B    #'0',D1
           BNE.S     N_HASH_0
           MOVEA.L   CHAN_0,A0
BACK_HASH  MOVEQ     #0,D0
           RTS
BAD_HASH   MOVEQ     #ERR_BP,D0
           RTS
N_HASH_0   CMPI.B    #'1',D1
           BNE.S     BAD_HASH
           MOVEA.L   CHAN_1,A0
           BRA.S     BACK_HASH
NOT_HASH   MOVEA.L   A0,A5
           QDOS      IO_OPEN,2
           TST.L     D0
           BEQ.S     GOT_IT_CO
           CMPI.L    #ERR_NF,D0
           BNE.S     UP_CREEK
           MOVEA.L   A5,A1
           MOVEQ     #3,D0
           BSR       NEW_HEAP
           MOVEA.L   A1,A4
           MOVEA.L   DEFAULT,A2
           MOVEA.L   A5,A3
           BSR       APPEND_S
           MOVEA.L   A2,A0
           QDOS      IO_OPEN,2
           TST.L     D0
           BEQ.S     GOT_IT_CO
UP_CREEK   MOVEA.L   A5,A0
GOT_IT_CO  RTS
```

In the final part of this series J. Pilkington describes the interface to the plotter's drive motors and some simple software.

DIY PLOTTER

The design of the interface between the BBC micro and the two stepper motors of the plotter is greatly simplified by the use of a dedicated IC which ensures that the motors receive the correct drive pulses. The stepper driver circuit is shown in **Figure 1** – two such circuits will be required. The interface is based on the SAA1027 IC which drives the phases of the motor in such a way that the axle of the stepper rotates by a precisely determined angle for every pulse applied. The step angle depends on the design of motor used and in the case of the motors the DIY plotter this is 79 degrees.

The stepper motor interface is powered from a 12V supply that also provides power to the motors. At this voltage the pulse bit (pin 15) and the direction bit (pin 3) are logic high for voltages between +7.5V and 12V and logic low for voltages in the range 0V to +4.5V.

Some form of logic conversion from the 5V logic of the BBC micro output port will be necessary. This conversion is accomplished by the two transistors Q1 and Q2. Note that these devices also have the effect of inverting the logic levels as output by the computer.

The stepper motor driver IC requires the following signals from the BBC computer.

Pulse Bit: each pulse must be at least 30mS long and each step angle of the motor will begin on the positive edge of the pulse.

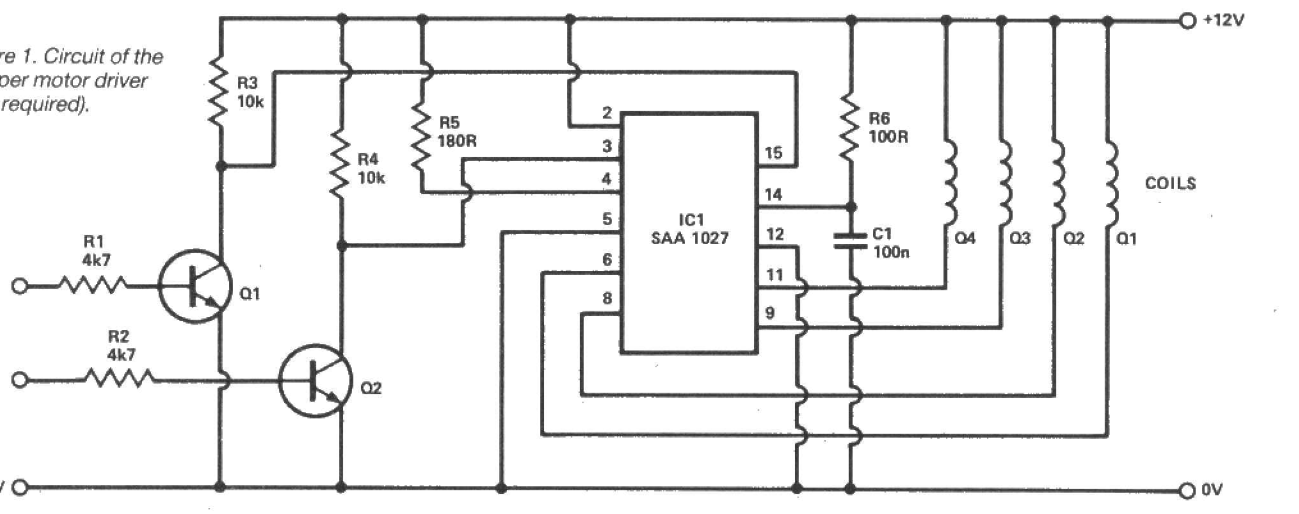


Figure 1. Circuit of the stepper motor driver (two required).

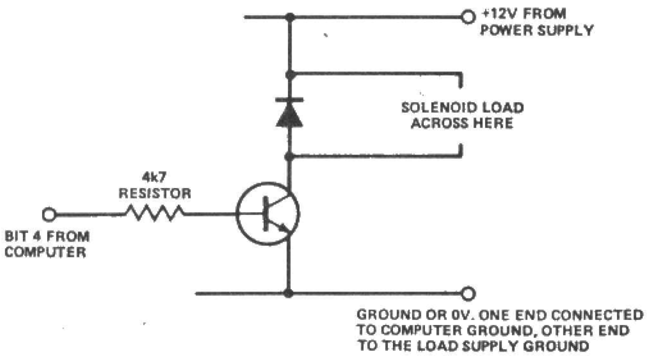

+12V FROM POWER SUPPLY
SOLENOID LOAD ACROSS HERE
4k7 RESISTOR
BIT 4 FROM COMPUTER
GROUND OR 0V. ONE END CONNECTED TO COMPUTER GROUND, OTHER END TO THE LOAD SUPPLY GROUND

Figure 2. Solenoid interface circuit.

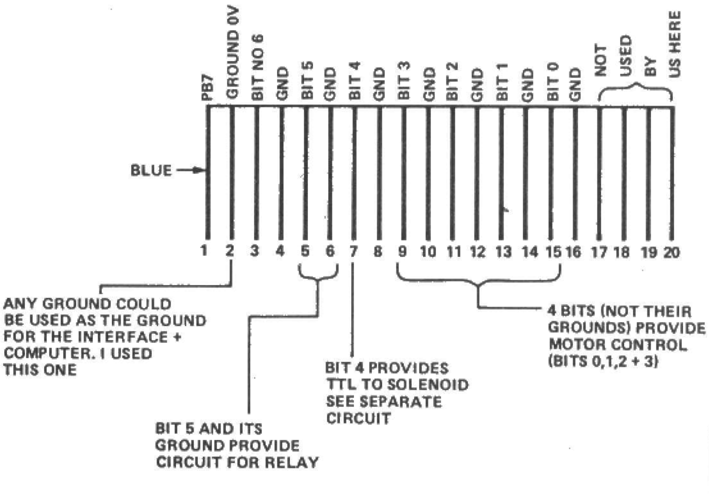

PB7
GROUND 0V
BIT NO 6
GND
BIT 5
GND
BIT 4
GND
BIT 3
GND
BIT 2
GND
BIT 1
GND
BIT 0
GND
NOT USED BY US HERE
BLUE
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
ANY GROUND COULD BE USED AS THE GROUND FOR THE INTERFACE + COMPUTER. I USED THIS ONE
BIT 5 AND ITS GROUND PROVIDE CIRCUIT FOR RELAY
BIT 4 PROVIDES TTL TO SOLENOID SEE SEPARATE CIRCUIT
4 BITS (NOT THEIR GROUNDS) PROVIDE MOTOR CONTROL (BITS 0,1,2 + 3)

Figure 3. User port connection details.

Direction bit: a logic high at the input of the SA1027 will cause an anti-clockwise rotation of the stepper while a logic low will initiate a clockwise rotation. These rotations are as viewed from the shaft of the stepper.

Pin 2 of the stepper IC is tied to logic high, this allows phase switching to occur.

Figure 2 shows the circuit of the pen solenoid interface. The transistor Q1 converts the logic level output of the BBC micro to a 12V signal capable of energising the solenoid. The diode prvents any back EMF – generated when the solenoid is operated – from damaging the transistor.

Figure 3 shows the connections between the plotter's interface electronics and the BBC micro. Bits 0, 1, 2 and 3 are used to control the stepper motors. Bits 0 and 3 provide the direction bits for the two motors while bits 1 and 4 are the motor pulse signals. Bit 4 is used as the solenoid control line. Should the plotter be fitted with an optional reed relay to detect the 'home' position, this can be connected to bit 5.

The software

The software to drive the DIY Plotter is, in concept, quite straightforward. What follows is an outline of the way in which simple control over the motor can be achieved, leaving you to explore more sophisticated software interfaces to the plotter.

The first task of the software is to configure bits 0 – 4 of the user port as output. This can be achieved with the following program line:

```
?65122=31
```

Control of the motors can then be achieved by sending the appropriate pattern of bits to location 65120. Motor 1 can be turned on by sending the value 1 to the port, motor 2 by the value 4 and both motors turned on by poking 5 to the port. Such commands should be written as part of a program loop, with the loop variable being the required number of steps. The loop should incorporate a line such as:

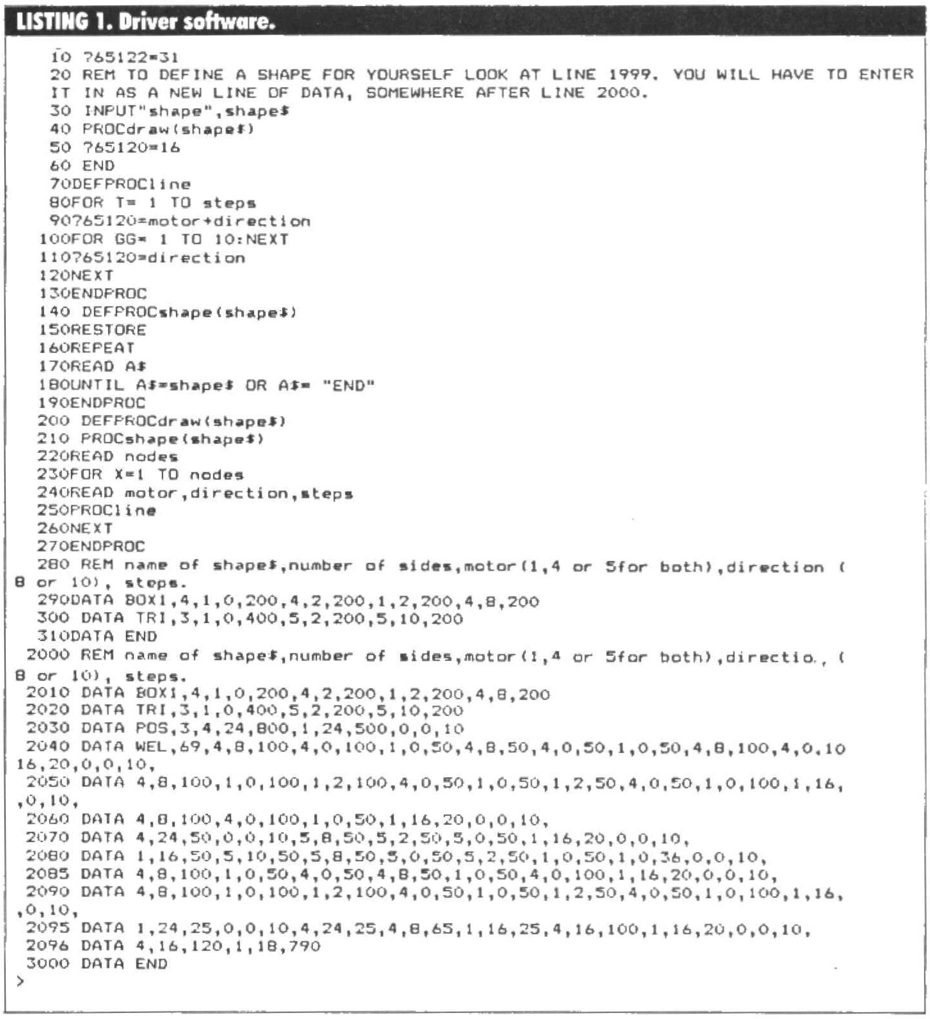
```
FOR D=1 to 10: NEXT
```

to introduce a short delay so that the step pulse meets the 30mS minimum duration demanded by the driver IC's spec. sheet. The loop must be terminated with a command that resets the pulse bit prior to it being set once more as the loop is entered on the next pass.

Before stepping the motors the appropriate direction bits should be set. Motor 1's direction bit is set by placing a value of 2 at location 65120 and motor 2's direction by outputing a value of 8.

The program shown in **Listing 1** shows how the simple concepts of driving the motors can be incorporated into a program that will allow shapes to be defined and drawn by invoking a single procedure. No doubt readers will be able to build on this program and conceive even more sophisticated methods of driving the plotter.

LISTING 1. Driver software.



```
   10 ?65122=31
   20 REM TO DEFINE A SHAPE FOR YOURSELF LOOK AT LINE 1999. YOU WILL HAVE TO ENTER
   IT IN AS A NEW LINE OF DATA, SOMEWHERE AFTER LINE 2000.
   30 INPUT"shape",shape$
   40 PROCdraw(shape$)
   50 ?65120=16
   60 END
   70DEFPROCline
   80FOR T= 1 TO steps
   90?65120=motor+direction
  100FOR GG= 1 TO 10:NEXT
  110?65120=direction
  120NEXT
  130ENDPROC
  140 DEFPROCshape(shape$)
  150RESTORE
  160REPEAT
  170READ A$
  180UNTIL A$=shape$ OR A$= "END"
  190ENDPROC
  200 DEFPROCdraw(shape$)
  210 PROCshape(shape$)
  220READ nodes
  230FOR X=1 TO nodes
  240READ motor,direction,steps
  250PROCline
  260NEXT
  270ENDPROC
  280 REM name of shape$,number of sides,motor(1,4 or 5for both),direction (
8 or 10), steps.
  290DATA BOX1,4,1,0,200,4,2,200,1,2,200,4,8,200
  300 DATA TRI,3,1,0,400,5,2,200,5,10,200
  310DATA END
 2000 REM name of shape$,number of sides,motor(1,4 or 5for both),directio., (
8 or 10), steps.
 2010 DATA BOX1,4,1,0,200,4,2,200,1,2,200,4,8,200
 2020 DATA TRI,3,1,0,400,5,2,200,5,10,200
 2030 DATA POS,3,4,24,800,1,24,500,0,0,10
 2040 DATA WEL,69,4,8,100,4,0,100,1,0,50,4,8,50,4,0,50,1,0,50,4,8,100,4,0.10
16,20,0,0,10,
 2050 DATA 4,8,100,1,0,100,1,2,100,4,0,50,1,0,50,1,2,50,4,0,50,1,0,100,1,16,
,0,10,
 2060 DATA 4,8,100,4,0,100,1,0,50,1,16,20,0,0,10,
 2070 DATA 4,24,50,0,0,10,5,8,50,5,2,50,5,0,50,1,16,20,0,0,10,
 2080 DATA 1,16,50,5,10,50,5,8,50,5,0,50,5,2,50,1,0,50,1,0,36,0,0,10,
 2085 DATA 4,8,100,1,0,50,4,0,50,4,8,50,1,0,50,4,0,100,1,16,20,0,0,10,
 2090 DATA 4,8,100,1,0,100,1,2,100,4,0,50,1,0,50,1,2,50,4,0,50,1,0,100,1,16,
,0,10,
 2095 DATA 1,24,25,0,0,10,4,24,25,4,8,65,1,16,25,4,16,100,1,16,20,0,0,10,
 2096 DATA 4,16,120,1,18,790
 3000 DATA END
>
```

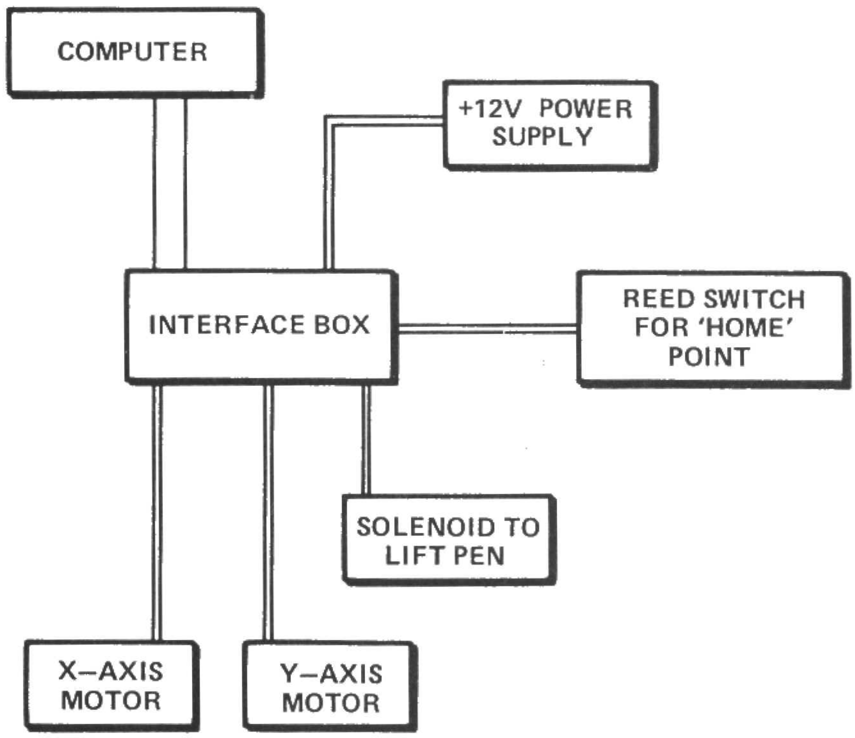


Figure 4. Block diagram showing the relationship between the various sections of the project.

The season's computer accessory is a modem and micro users are becoming spoilt for choice. But which features are really worth having and which are gimmicks? Jonathan Frost looks at nine popular models and gives a six point guide to modem buying.

logging on in six easy lessons

When Penrith-based OE Ltd went into receivership earlier this year following the demise of its major distributer Prism Microproducts, it marked the end of a significant period in data communications in the UK.

OEL was arguably the first company to predict the recent surge in the popularity of microcomputer communications and as early as 1981 were manufacturing the cheap acoustic modem which made possible the launch of Micronet 800 and was to be the foundation of their later success.

The traditional modem companies with long and distinguished records of product excellence and customer service had not addressed the home micro market at all. Perhaps they failed to spot the trends but the more likely reason was their inherent incapacity to adopt a radically different approach for a radically different market (added to more than a modicom of professional disdain for the cheap micros). This would certainly account for their conspicuous inactivity since the home micro data communications boom began.

So it was left to the new small and agile companies like OEL and latterly PACE and Miracle Technology to carve up the market.

Despite making great strides towards the establishment of an independent dealer network, OEL, funded by the Yorkshire bakers Warburtons (that's what you call diversification), could not survive the shock waves caused by Prism's implosion. They called in the receiver some nine weeks later with considerable stocks of modems and comms packs to dispose of.

They left behind a clutch of young outfits supplying cheap sophisticated equipment to micro enthusiasts.

The result is confusing for the potential modem purchaser. The volumes of liquidated stock, much of it bought up by Modem House of Exeter, mean that Prism and OEL equipment is currently very cheap. While stocks last this places manufacturers of other more powerful modems at a temporary disadvantage.

There are six essential points to remember when buying a modem:

1. It is a false economy to buy a simple modem if you envisage that your interest in data communications may soon outstrip its facilities. Modems are generally not upgradable and are difficult to sell secondhand.

2. Terminal software: The most crucial aspect of your purchase is not the modem itself but its software. Make sure there is a good software package to accompany the modem which makes good use both of the modem and your micro.

3. Shop around: A number of promotional gimmicks are currently being employed because competition is fierce. Good discounts are available from some dealers and at some computer shows in addition to free subscriptions to network services like Micronet 800.

4. Cables: A very sore point amongst computer users. If a cable is not included in the price of the modem or terminal software it will probably cost you between 10% and 15% extra.

5. BABT Approval: All modems intended for connection to the UK telephone network must have full approval from the British Approvals Board for Telecommunications and bear the famous green sticker. Don't take any rubbish from salesmen – if the modem they're trying to sell you doesn't have a green sticker then it's *not* approved and therefore illegal to plug in at home. Even if the modem is "awaiting BT approval" that's no guarantee it will get it. Neither does it guarantee that yours is not an older model the illegal status of which will remain unaffected by subsequent approval for the new version.

6. Bell tones: Many modem manufacturers indirectly advertise their products as allowing you to connect to enormous American databases using Bell frequencies. It is illegal to use Bell standard modems on the UK telephone network unless a special guard frequency is inserted because they can play havoc with the signalling between British Telecom's AC9 group switching centres. Modems with an unguarded Bell 103 originate tone will be approved by BABT only if they have their Bell connections disabled at the factory.

'Don't take any rubbish from salesmen – if the modem they're trying to sell you doesn't have a green sticker then it's *not* approved and is therefore illegal'

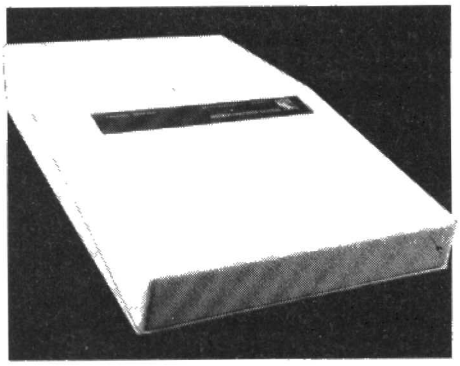
BUYER'S GUIDE

The following is a useful but by no means exhaustive list of low cost modems designed for micros. Most manufacturers have modems which will be ready "any day now" but we mention here only those which are on the street, not on the drawing board. They can be broadly categorised as single-standard V23 modems used primarily for Prestel, or multi-standard V21/V23 modems which can additionally support communication at 300 bits per second for a much wider range of services. They are all based around one of only a handful of modem chips and consequently there is little to choose between their performance. Guide prices quoted are approximate and include VAT. Features are summarised in the table on the right.

	Guide price	300/300	Prestel	Reverse Prestel	1200/1200	Auto-dial	Auto-answer	Software included	Software available for				BABT approved
									BBC	Comm 64	Spectrum	QL	
Acorn Prestel Adaptor	115		●			●		●	●				●
Commodore Modem	100		●		●	●		●		●			●
Demon	84	●	●	●	●	●	●	●	●				
Bright Star	180	●	●	●	●	●	●	●				●	
OEL/Prism Modem 1000	50		●		●				●	●			●
OEL/Prism VTX5000	50		●		●			●			●		●
Pace Nightingale	130	●	●	●					●				●
Miracle WS2000	150	●	●	●	●				●				●
Tandata Q-CON	180		●		●	●	●	●				●	

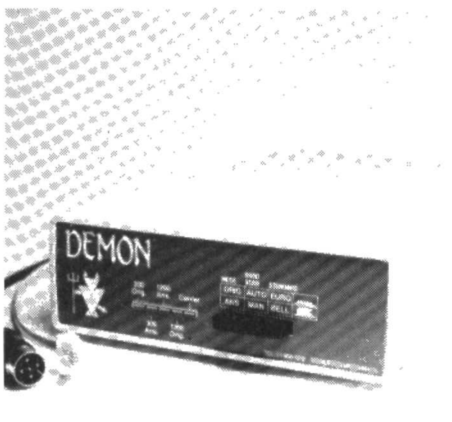
ACORN PRESTEL ADAPTOR

A smart but overpriced and now rather overtaken single-standard modem dedicated to the BBC Micro. The terminal software, written by Soft Machinery, is very good and is supplied on ROM (included in price).



DEMON

An excellent little device, this state-of-the-art modem is imaginatively designed and destined to take a large slice of the market. Although not yet BABT approved it is selling well and has received rave reviews ever since it was released. Currently there is only software available for the BBC Micro but they're working on others. The Demon is a slim multi-standard auto-dial auto-answer modem with auto baud-rate scanning (ie it can automatically detect the incoming data rate and configure itself accordingly). The ROM software is particularly good and allows the user access to ROM routines from BASIC.



OEL/PRISM VTX5000

A classic piece of engineering, the VTX5000 is a single-standard device dedicated to the Sinclair Spectrum and contains Prestel software and the appropriate interfaces for connection via the expansion port. Although for use primarily with Prestel and for 1200/1200 bps user-to-user communications, it can be persuaded via soft overlays to connect to a variety of other systems like Telecom Gold and PSS (the packet switched stream system).

MIRACLE WS2000

The WS2000 is a multi-standard modem which has made quite a name for itself and for Miracle Technology especially among microcomputing professionals.

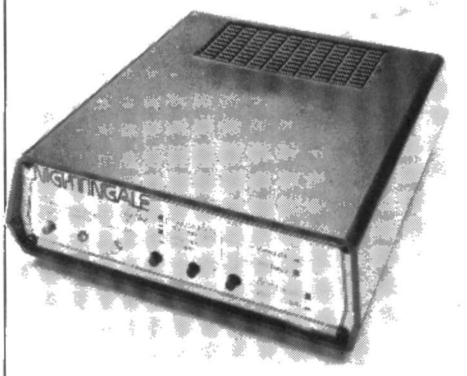
Providing much the same features as the PACE Nightingale but at high cost the WS2000 uses chunky rotary switches in place of the Nightingale's discreet push-buttons.

Software is available from Miracle for the BBC Micro, Atari and soon for the Commodore 64 (written to Micronet's new MUSTANG telesoftware protocol) and Spectrum.

PACE NIGHTINGALE

The PACE Nightingale is a popular multi-standard modem providing Prestel mode and 300/300 bps as well as 'reverse Prestel', that is, transmission at 1200 bps and reception at 75 bps enabling communication with any Prestel terminal in videotex format.

PACE are soon to release an auto-answer auto-dial board which will also allow complete configuration of the Nightingale. Very good terminal software, called Commstar, is available on ROM for the BBC. Software for other micros will be coming shortly.



OEL/PRISM MODEM 1000

The first direct connection modem from OEL, the Modem 1000 (or Telemod 2) is a single standard modem popularly used on the BBC Micro, Apple and Commodore 64.

Originally sold at just under £100 it was not a phenomenal success in sales terms (highlighting the problems of selling modems to owners of very cheap micros) but at just under £50 from Modem House it should do rather better.

TANDATA Q-CONNECT

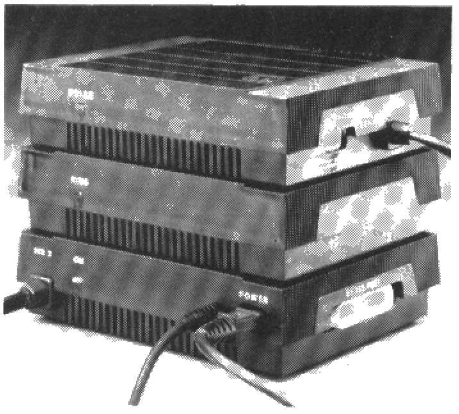
One of OEL's last projects was a complete communications system for the Sinclair QL. Tandata have bought the rights and have begun to market the system under the same name.

Q-CONNECT comes in three stylish modules which interconnect in a stack formation and all are available separately.

Q-CON provides an intelligent serial interface to the QL's serial port which, on its own is hopelessly inadequate for data communications. Terminal software, written by Scicon, is supplied on microdrive cartridge.

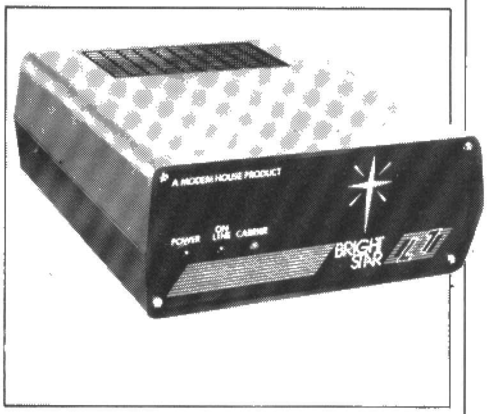
Q-MOD is a simple V23 modem for Prestel and 1200/1200 bps half-duplex. Q-CALL is an optional auto-dial auto-answer module for Q-MOD.

Although the software supports modes other than Prestel Q-MOD does not so it won't be long before we see a number of multi-standard modems designed to fit into Q-CON which exploit its software more fully. In fact, the Q-CONNECT system is very expensive considering its limited range of applications.



MODEM HOUSE BRIGHT STAR

The first Modem House 'own brand', the Bright Star is an intelligent multi-standard modem with built-in auto-dial and auto-answer, auto baud rate scanning and full flow control. Software is currently only available for the QL but BBC Software is under development. This modem will no doubt take the lion's share of the QL market and will certainly compete with the Demon for the more affluent BBC user.



COMMODORE MODEM

Of all the modems in this list the Commodore modem, dedicated to the Commodore 64, is the only one based upon rather outdated technology. Although doubts were expressed in the early days about their capability over long distances they now seem to perform quite adequately both on Prestel and in user-to-user mode at 1200 bps.

The modem has neither switches nor front panel LEDs and connection to the Commodore 64 is via the games cartridge port. Software for Compunet lies in ROM within the modem and includes a firmware identity which is used as an encryption key to prevent downloaded software from being run on any other micro.

This ROM can easily be overlayed from disk or cassette with terminal software for Prestel.

The Commodore modem is overpriced at £100 even if you do take account of the "free years subscription" to Compunet included in the price. Starting in September early subscribers will have to start paying for their membership so it will be interesting to see the levels of migration to Prestel-based services.

SUPPLIERS

Acorn Prestel Adaptor: Acorn Computers 0223 210111. Commodore Modem: Commodore Business Machines 0536 205252. Demon: Demon Ltd 01 930 1612. Bright Star, OEL/Prism Modem 1000, OEL/Prism VTX5000: Modem House 0392 69295. Pace Nightingale: Pace 0274 739306. Miracle WS2000: Miracle Technology 0473 50304. Tandata Q-CON: 06845 68421.

NET NEWS

Micronet to expand after sale of stock

● Micronet's acquisition of a "seven figure sum" investment from Bell Canada is likely to produce more interactive services like the new and successful Gallery and Chatline areas.

The cash injection gives Canada's most profitable concern a 25% shareholding in Telemap, Micronet's parent company. In the same reorganisation, British Telecom's long-standing involvement in the company has been formalised by a 20% stake – in exchange for the abondonment of a prior agreement under which BT would take a part-share of future Micronet subscriptions.

East Midland Allied Press (which also publishes *E&CM*) retains the major shareholding of 55%, and will be delighted at an investment which values Micronet at a fugure much higher than its £1m development cost.

Micronet's share of the money is likely to go on interactive services similar to its Chatline area – regularly the most accessed feature on Prestel.

But the bulk of the cash will be used on Telemap projects to run alongside Micronet – the largest of which is "Buttons", a compendium of games and quizzes open to anyone with a numeric keypad and Prestel account (no access to the Prestel Microcomputing area will be necessary).

Curiously, the collaboration of BT and Bell Canada on the new Telemap board follows the former's purchase of the Mitel Corporation – which brought the two telecoms giants into direct competition in the North American market.

Autumn will bring more MUD

● MUD supremo Richard Bartle has been telling us about his collaboration with BT and Micronet on two new multi-user games.

From the autumn, Richard expects MUD 2 (assuming that the Essex set up was MUD 1) to be running on a BT VAX in London. Between 6pm and 9am, 100 lines will be open to the improved adventure game, with access available by PSS and – as soon as it opens – via BT's expanded local call network.

"It'll cost around £2 an hour," Richard told us, "which is too high, in my opinion. But there'll be discounts for early subscribers and for people who buy lots of games credits in one go."

The Micronet game, adapted by a commercial software house, should be on-line early next year, "Though there are some fearsome technical problems – notably that MUD scrolls and Prestel doesn't."

The new versions of MUD will be transportable, says Richard, and he's hoping that sections can be lifted from the full game and implemented on home micros.

Telecom Gold turns blue

● BT Gold's bulletin board was closed for part of last month after complaints from Euro MP Barry Seal that the service was being used for a "sex-dating service like the smutty magazines sold in backstreet shops."

Mr Seal told us that he's a subscriber to both Micronet and BT Gold – which he uses for rapid messaging between his West Yorkshire constituency and the Euro-offices in Brussels. It was a "particularly sick joke" about the Brussels football disaster which alerted him to the Gold goings-on.

Regular Gold users will have their own tales to tell about facilities available through the subscribers' bulletin board. Perhaps the most notable contact is the famous Sheila, who sends full details of her wares – a "night of pleasure" costs £200, while more esoteric services requiring the use of leather and belts are available for £500 – to anyone writing to her Gold account.

For Mr Seal, however, such facilities came as a revelation, and he was swift to complain both to the Home Office and to BT direct.

A Gold executive confirmed that discussions are underway to clarify where legal liability resides for the contents of public access systems – an area with little legislation or legal precedent.

But Mr Seal has no doubts – BT, he feels, must take full responsibility, just as an editor does for his newspaper. Other Gold users, however, believe that the system should be regarded as a "passive carrier" like the normal telephone system, with the responsibility for offensive messages residing with users, not with the network they employ.

Micronet resists censorship attempts

● Micronet's Gallery service has swing into action, and already boasts a gay contacts board, a hatful of software, film and record magazines, and – for one brief period – the ravings of an anonymous hacker.

The Gallery's gay switchboard has raised the well-worn issue of how much responsibility Prestel and its information providers should take for the input of end users.

A couple of subscribers have already asked that Micronet remove the gay area, but the magazine's Technical Manager Mike Brown insists that this won't happen.

"All we ask," Mike told us, "is that Gallery editors stay within the requirements of their Prestel customer contract. This demands that subscribers shall not use Prestel for obscene, libellous or otherwise offensive material.

"Provided Gallery frames stay within the law, and within these modest Prestel requirements, we won't interfere."

Not even if the National Front took Gallery pages? we wondered.

"That's easy," said Support-the Miners Mike. "The National Front *could* use Gallery, provided it didn't break the law by inciting racial hatred."

Proof that Micronet isn't unshockable came with the new area's first hacker.

Using the passwords of a legitimate Gallery editor, the interloper exploited the fully-automated updating procedure to fill four frames with foul-mouthed boastings about his ID-stealing abilities.

Fortunately, before most subscribers were aware of them his meanderings had been spotted by Micronet editorial and – under the aforementioned anti-obscenity regulation – had been rapidly excised.

Modems on the cheap from failed Protek

● Protek, whose acoustic modems introduced many a future hacker to the wonderful world of comms, has called in the receiver.

The company never managed to move on from its increasingly outdated device, and after the demise of more energetic concerns its departure will surprise no-one.

But the good news is that the receiver (shades of Prism, here) is in possession of dozens of the whistling wonders, and will be happy to hear bids for any number down to one.

If you're nostalgic for the string and chewing-gum days of telecoms – or, more seriously, if you appreciate the portability of acoustic modems – call 0506 415353 and name your price.

NET QUESTIONS

To solve all (or most!) communications problems write to NET QUESTIONS, E&CM, Priory Court, 30-32 Farringdon Road, London EC1.

QUESTION: What is a 'carrier frequency'?

Data is transmitted over the telephone network in the form of audible frequencies. Normally only two frequencies are used – one for binary 1 and one for binary 0. There are more expensive modems which use four frequencies, for 00, 01, 10, 11 allowing data to be sent twice as fast over the same telephone link, but such modems are expensive. Each character has a unique binary code and a modem will convert this code to a sequence of frequency changes for transmission. In the interval between characters the frequency remains that for binary 1 ('idle') and is called the carrier frequency, the start of each character being heralded by a single binary 0.

QUESTION: What is 'auto baud-rate scanning' and how does it work?

Auto baud-rate scanning is a technique employed by modern intelligent modems enabling a user to connect to a remote computer system without prior knowledge of the data rates required. Modems attached to large computer systems normally operate at only one receive and one transmit data rate (eg. 300/300, 1200/75, 1200/1200). For each of these data rates the carrier frequencies have been carefully selected by international agreement to avoid confusion between terminal and host and to allow data to pass in both directions at the same time ('full duplex'). When you dial one of these host modems it will automatically answer the call and send out the appropriate carrier frequency for the data rate it wishes to use to transmit to your terminal. Your modem must listen for this carrier and respond by supplying its own carrier to the telephone line. With both modems 'singing' to each other like this the connection is established and data transfer may commence.

An auto baud-rate scanning modem will listen for each of the carrier frequencies it is likely to encounter in turn until one is found and it responds by returning the carrier frequency which the host expects from the originating terminal. All this goes on quite independently of the terminal microcomputer itself; when the connection is established the micro is informed that data for transmission to the host may now be handed to the modem. The speed of this data input to the modem is likely to be high and is quite independent of the data rates on the telephone line – the modem will simply buffer output from the micro for transmission and also buffer the incoming data from the host.

The consequence is that communications software can be written much more easily since there is no need to reconfigure your serial port to adapt to the host system (you just squirt your data out and get data back) and you can assume that the transfer is full-duplex even if it's not (the modem will handle the flow control for you).

Even cleverer 'adaptive' modems will adopt the highest data rates possible on any data link and will automatically switch by mutual agreement to lower data rates whenever the signal quality deteriorates to the point where data corruption reaches unacceptable levels.

QUESTION: What is 'parity' and how do I include it in my data comms software?

Parity is a mechanism which allows the detection of simple errors in single characters transmitted over a data link.

An extra bit, the parity bit, is added to the character code before transmission such that there is an even number of binary 1s in the whole "word". For example, the ISO7 code for the letter C is 67 which is 1000011 in binary; the parity bit here must be 1 to give an even number of binary 1s in the whole word. So C would be transmitted as the eight bit word 11000011. Similarly, the letter A 10000001 would become 01000001.

If a terminal finds that there is an odd number of binary 1s in a received word then some corruption of the character has occurred. The terminal has no way of deducing what the original character was and so would normally ask the host for a retransmission.

On Prestel, however, the integrity of the data (except for telesoftware) is not so critical and terminals are required only to display a small 'blob' (Prestel code 127) in place of any erroneous character.

There are a number of algorithms for checking and inserting parity and you may find that your hardware does it for you. If not, here are some fairly simple methods which you can use from BASIC. If your BASIC doesn't support EOR, the exclusive-OR operation, then you'll have to use an alternative:

```
A EOR B = ((NOT A) AND B)
OR (A AND (NOT B))
```

To check even parity just pick up an incoming word from your serial port, we'll call it k, then perform the following:

```
p = 16*k EOR k
k = k AND 127
p = 4*p EOR p
p = 2*p EOR p
p = AND 128
```

where p is an integer.

If p is zero then the code k should be passed to your screen handler otherwise a parity error has been detected.

To insert even parity on an outgoing character c, use:

```
C = c AND 127
p = 16*c EOR c
p = 4*p EOR p
p = 2*p EOR p
c = c + (p AND 128)
```

The Prestel system does not check parity on any incoming characters but the Prestel Terminal Specification insists that you include it. Other viewdata systems will only accept characters without parity errors so it's a good idea to program it in anyway.

QUESTION: Some telephone systems use a signalling system referred to as DTMF. What do the initials stand for and is it possible to use a standard modem with a 'phone system using DTMF.

The initials DTMF refer to the Dual Tone Multi Frequency signalling system used in some PABX systems. The term refers to the way in which the system originates a call. The BT trunk network uses a line pulse dialling system in which the digits of the number being dialed are signalled by rapidly seizing and releasing the line. This system dates back to the time when most 'phones used a mechanical dial mechanism. In a DTMF system dialling information is communicated via a set of tones with each of the digits 0-9 associated with a specific frequency. This system is far more efficient than line pulse dialling and is found in most modern digital equipment.

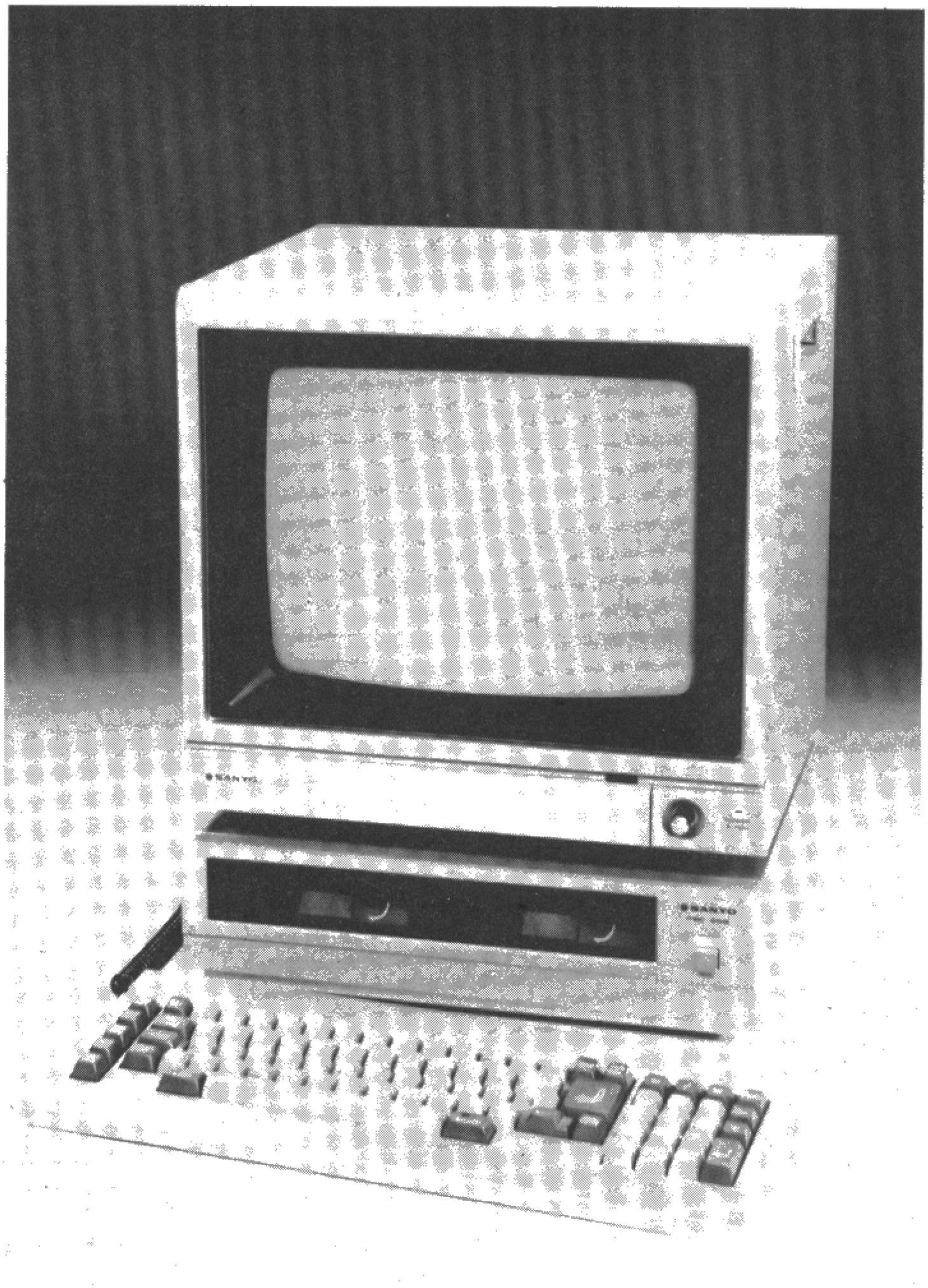
A DTMF system can be used with any standard modem without any modification. The only point to note is that any auto-dial system will not function as these units use a relay to pulse the 'phone line – this will have no effect on a DTMF system. Note that for reasons of compatibility with existing equipment, even BT's modern digital exchanges operate on a line pulse dialling system as far as the public is concerned although BT will use digital tones for other purposes.

NET NUMBERS

PLEASE NOTE

The correct number for Forum 80 Hull is 0482 859161, Forum 80 Spa is open between 22.30 and 00.30 **not** 13.00 to 00.00 as previously stated. We apologise for any inconvenience caused to those concerned.

A sixteen-bit computer using proven technology, with two disk drives, MS-DOS, and a wealth of bundled software all for £800? Simon Craven saw Sanyo and decided this was more than just for business.



SANYO MBC 555

The main operating system currently used on business micros is MS-DOS, or its close relative IBM PC-DOS, which run, in various versions, on machines based around the Intel 8088 and 8086 processors. Until recently MS-DOS has made little impact on areas of personal computing outside the business world – indeed, two recreational machines conforming to this general specification, the IBM PC Junior and the Advance 86, have failed to capture the imagination of the buying public.

Now, however, there does seem to be a swing towards the use of MS-DOS in wider applications. The educational computing market is catered for by the Apricot FIE and the Research Machines Nimbus, a powerful 80186 computer, and the supporters of MS-DOS in this environment claim it gives students a good background in the kind of computers they will need to use in the real world. Another factor is the increasing availability of high quality software at relatively low cost, to suit educational and private buyers.

A computer which seems almost tailor-made to help the growth of MS-DOS into new areas is the Sanyo MBC500 series. The basic machine, the MBC550, has been around for some time, but it has evolved from a rather uninteresting specification with one 160K 5.25" floppy disk drive to the point where the latest version, known as the MBC555+, can offer two double-sided, double-density 80 track 720K drives, giving a healthy 1.4 Mb total of mass storage. This is twice what a floppy-disk IBM PC can offer.

This capability is remarkable given the price. The top-of-the-line MBC555+, including 128K of RAM, parallel printer interface and a monochrome monitor can be had for around £1100 plus VAT, which compares with a typical discounted price of £1800 for a 2 x 360K IBM PC configuration. Throw in a decent dot-matrix printer and your complete Sanyo system would cost under £1500 including VAT. If you can get by with less storage, a similar complete system including printer and VAT but with 160K drives replacing the 720K units would cost under £1300 (ie. the basic 555 with monitor can be bought for £800 excluding VAT).

A considerable amount of free software is included with the machine, though as always, this is only to be taken into consideration if it is the software you would have bought anyway. Apart from the operating system utilities and Basic, you get Micropro's family of applications programs. Wordstar and Mailmerge are two that everyone seems to know, but less familiar are Datastar and Reportstar, a database management system and report generator, or Calcstar, which is a fairly basic, though straightforward spreadsheet package.

'As a low-cost machine for serious word-processing, programming in the many languages available under MS-DOS, or other applications where a tough MS-DOS machine is needed, the Sanyo is very attractive'.

Some Sony dealers have advertised the MBC550/555 as an IBM compatible machine, but this claim seems to be based mainly on the similar operating system and general specification. Depending on what combination of disk drives you have, the Sanyo can read and write IBM PC disks of 160K or 320K (single-sided under DOS version 1, or double-sided under DOS version 2), so exchanging data between this machine and any of the IBM PC clones is simplicity itself.

Any software which is written entirely legally under PC-DOS or MS-DOS rules, and which does not require graphics, can be expected to work on the Sanyo. This includes quite a bit of public-domain software available via the IBM PC User Group and other sources. Anything which uses graphics is in a no-go area, as is anything which addresses the hardware directly. This means that Lotus 1-2-3 and the Microsoft Flight Simulator won't work unless you buy an IBM compatible graphics board.

Quite a lot of the BIOS (Basic Input Output System) entry points are the same as those of the IBM, so the Sanyo is fundamentally closer to IBM compatibility than the ACT Apricot, which also has physically incompatible 3.5 inch disk drives.

Any operating system batch file programs will work satisfactorily, as will many programs written in Basic.

The Basic interpreter supplied is not quite up to the standards of IBM BASICA or Microsoft GW-BASIC, but it has many similarities. It is obviously descended through the Microsoft family tree, and resembles a slightly simplified version of the current GW-BASIC, rather like the simplified Basic IBM puts in ROM on the PC.

The internal coding of stored Basic statements is not the same, so Basic programs cannot be transferred directly from one interpreter to another. The procedure is to save the program as an ASCII text file (SAVE "FILENAME.EXT"A) and then load it on the other machine. This works from IBM to Sanyo and vice versa.

You still aren't guaranteed to be out of the woods, though, and if the program uses any of the IBM-only statements some modifications will be necessary.

Physically, the Sanyo is extremely attractive. It looks more like hi-fi or video equipment than the rather poorly-finished computer equipment we usually put up with.

The main system unit, containing the CPU, the RAM and the disk drives, is particularly neat. Measuring 15 by 15 by 4.5", it is about the same size as a slim record turntable. The desktop footprint, weight and general bulk are all much smaller than those of most desktop MS-DOS machines, which is an advantage even if you are working in an office.

Around the back are the keyboard connector and the parallel printer interface, thankfully a standard Amphenol connector – not one of the strange connectors that Japanese equipment often uses. The power switch is on the front, so you can reach it without going through ridiculous contortions. Two display outlets are provided – composite video for the monochrome display, and linear RGB via an eight-pin DIN socket for colour addicts. Both displays can be operated at one, which is not as common among business computers as it is among home machines.

Fortunately the Sanyo does not follow the IBM practice of not fitting a proper hardward reset switch. A reset is initiated by pressing a little switch hidden in the left edge of the keyboard.

Inside, the Sanyo is pretty crowded. The disk drives take up much of the space inside the box, and there is no room for the row of expansion slots which made the IBM PC (and before it the Apple II) such a great machine for playing around with the hardware. If you really want to get at the guts of the machine, you could try experimenting with the 62-pin bus on the main PCB. This gives access to the address and data busses, the RAM and I/O signals.

RAM can be taken from 128K to a maximum of 256K, by adding 16 4164 RAM chips to the main board.

The main processor is an Intel 8088, not the more advanced 8086 which has a 16-bit data bus. The clock rate is a sluggish 3.6MHz, making this the slowest machine in the IBM PC class – most PC compatibles boost speed to double the IBM's unexceptional 4.77MHz.

An Intel 8087 maths co-processor can be added to an empty socket next to the 8088. This greatly speeds up floating point operation if the software (such as many compilers) is configured to take advantage of it.

The keyboard is fairly standard, with the numeric pad doubling as the cursor pad. Only one row of function keys is fitted. The keys are very pleasant to use, and one area of superiority over the IBM is the presence of LED indicators to indicate when the caps lock or shift lock are engaged.

The standard monochrome monitor is a classy unit with similar styling to the Sony Profeel range. One slight defect is that no tilt facility is available, but the display is excellent in eighty-column text mode.

Something that many *E&CM* readers would require is a serial communications card. This is available as an extra, giving baud rates between 110 and 4800. Communications software is needed to get the most from the port, but an assembly listing in the manual tells you all you need for basic setting up. The serial controller is the Intel 8251A.

As a low-cost machine for serious word-processing, programming in the many languages available under MS-DOS, or any other application where a tough, reliable MS-DOS machine is needed, the Sanyo is very attractive. The main disadvantages are its low speed and lack of hardware expandability, but the exceptionally low cost and flexible disk storage options more than offset these drawbacks.

DATAFILE SANYO MBC 555

Processor	Intel 8088 3.6MHz
RAM	128K expandable to 256K
Mass Storage	160K, 320K or 720K double disk drive options
Operating System	MS-DOS V1 (160K drives) MS-DOS V2 (320K, 720K drives)
Keyboard	Full travel. Includes
mnemonic keypad	and five function keys
Graphics	640 x 200 pixels. Colour card as standard
Interfaces	Centronics, composite video, RGB, 62-pin expansion bus
Bundled software	BASIC, Wordstar, Mailmerge, Spellstar, Datastar, Reportstar, Calcstar
Further information	Sanyo 0923 46363

THE FIRST SPECTRUM WORDPROCESSOR IN FIRMWARE

PART FIVE

Three routines

Richard Sargent has added three new facilities to his Spectrum wordprocessor: hex/decimal number conversion, sorting, and large character printing.

This is the penultimate part of the Spectrum wordprocessor in ROM, and we look at the three routines which are enhancements to the main body of code: SORTING, LARGE-CHARACTER PRINTING and NUMBER CONVERSION. The code for the latter is produced in full since it can be incorporated into any Z80 computer.

Converting numbers between Hexadecimal and Decimal is not too easy using Sinclair BASIC, and it would be useful if the wordprocessor did the job. Quite a lot of the necessary code is already in the program. For example, to display the BYTES FREE status value a pure binary number in the HL register is converted to a five digit decimal number, represented on the screen as ASCII digits. It doesn't take much, therefore, to make the wordprocessor do a conversion "on demand", but so what? If you have one of those smart solar-powered CASIO calculators, you can undertake any number of conversions in any number of ways, at least until the sun goes down! What Spectrum WP does, however, is to alter numbers in the text, and there need be only one or alternatively, hundreds of numbers: the necessary conversions are all done automatically.

Listing 1, which is this month's software offering, shows the code. The routine has been adjusted to enable it to run on Z80 micros generally, and it's assembled to fit into the Spectrum's memory by sitting in front of the 2K RAM WP which starts at 8000H.

Maths routines

The key routines are the two marked **MATHS1** and **MATHS2.** These are general sub-routines which can be used whenever number conversion is required in a machine-code program. MATHS1 is supplied with a hexadecimal number in the form of a positive 16-bit binary value in register HL. Register DE is set to point to a spare area of RAM, five bytes in size. On exit from the routine, the ASCII/decimal conversion will be found in the five bytes (eg: ST1 to ST5) with the first byte (eg: ST1) holding the most significant digit. Leading zeros are not suppressed at this stage, though they can be later: the code presented does not suppress zeros for reasons which will shortly become apparent.

MATHS2 converts the other way. On entry to the routine the address of the first ASCII/decimal byte should be held in the HL register. There are two exits from the routine. Successful conversions leave via **JR C DONE** or **JR NC DONE,** with register BC holding the 16-bit binary number. Unsuccessful conversions leave via one of the many **JR C BAD** routes. Pay no heed to what happens at label DONE and at label BAD: the code there relates to the whole program, not to the workings of sub-routines MATHS1 and MATHS 2.

Conversion in situ

The remaining code of Listing 1 changes the numbers within the wordprocessor text. To do this, the numbers must be marked (otherwise the program won't know which ones to change). It is also helpful to expand those decimal numbers destined to be changed to five-digit length and hexadecimal numbers ear-marked for change to four-digit length. The conversion program can then work on the text file without having to alter the length of it to accommodate the new numbers! This is the short cut which makes it easy to apply the conversion program to virtually any wordprocessor text. **Listing 2** shows the process in action, based on an imaginary piece of text.

The program is of greatest value in situations where the wordprocessor's "search and replace" command is used to place markers in front of numbers. As shown, the marker for hexadecimal conversion starts at the cursor position in the text and continues until either the "stop-" marker or the end of text is reached. The stop marker is the @ symbol, but this symbol, and that of the other markers can be easily changed.

The program is RUN by a **CALL** to **CONVERT** at 7FDBH. Here, after saving the registers, the program picks up the address of the cursor in the text, LD HL,(CURSOR) and proceeds for symbol-markers. If you can't find the cursor storage-location for your wordprocessor, then you should load HL with the start of your text area. Use LD HL,xxxx not LD HL,(xxxx).

To use the program with the 2K RAM WP it is only necessary to patch the new routine's address into the system and allocate a command letter (X is spare) by which to summon it. Loading 80DF with 7F and 80E0 with DB does the trick.

Sorting

Sorting is the process of comparing and rearranging records in a file. Spectrum WP has the ability to consider part or all of its document/file as a record/file and will perform an alphanumeric sort on the records there, putting the items back into the document in ascending order. The normal workings of the wordprocessor are not impaired by this action, except that everything shuts down while the sort is in progress!

Bubbles

The bubble or ripple sort has the dubious honour of being about the slowest type of sort that there is. However, it is also one of the easiest to understand and to write, and for modest numbers of records (less than 500) it is not unbearably slow when written in machine code. The action of the bubble-sort has been well documented. It tends to be taught in schools, and it's also on the "Welcome" tape supplied with the Sinclair Spectrum. (It's the program where BASIC orders a hard of cards by rank.) The bubble sorting algorithm, put briefly, is to compare two *adjacent* records. If the two are already in the correct order, no action is taken, but if not, they are swopped around. The sort repeatedly looks at pairs of records until a complete pass of the record-file *without swopping* is achieved, and then the program knows it has finished. **Listing 3** shows four records from a hypothetical Video-Film list as they might appear in a printout.

Fields

Each record is composed of a number of separate parts or *fields,* of which the film title is only one. It is important that any of the seven fields represented can be selected as the sort field, so in order to sort the list by stock-number rather than by title, the routine would be instructed to examine field one. A sort by cassette type (VHS, Beta etc) would be done by choosing field number three. A non-alphanumeric symbol is used to distinguish the point at which one field ends and another starts and the carriage-return

LISTING 1. Number conversion.

```
00FB             ;
00FB
7F00                        ORG 07F00H
7F00                        ;LOAD 07F00H
7F00                        LIST 1
7F00
8823             CURSOR     EQU 8823H
0023             SYM0       EQU 23H;hash
005E             SYM1       EQU 5EH;up arrow
0040             SYM2       EQU 40H;at
00FF             SYM3       EQU 0FFH;end of
00FF                                   file
00FF             ;--------------------------

7F00 DDE5        MATHS1     PUSH IX
7F02 DD21287F               LD IX,DATA
7F06 3E2F        HD1        LD A,2FH
7F08 DD4E00                 LD C,(IX+0)
7F0B DD4601                 LD B,(IX+1)
7F0E C601        HD2        ADD A,1
7F10 ED42                   SBC HL,BC
7F12 30FA                   JR NC HD2
7F14 09                     ADD HL,BC
7F15 12                     LD (DE),A
7F16 DD23                   INC IX
7F18 DD23                   INC IX
7F1A 13                     INC DE
7F1B 0D                     DEC C
7F1C 20EB                   JR NZ HD1
7F1E EB                     EX DE,HL
7F1F 2B                     DEC HL
7F20 DDE1                   POP IX
7F22 C9                     RET
7F22
7F23 00          ST1        DB 0
7F24 00          ST2        DB 0
7F25 00          ST3        DB 0
7F26 00          ST4        DB 0
7F27 00          ST5        DB 0
7F28 1027        DATA       DW 10000
7F2A E803                   DW 1000
7F2C 6400                   DW 100
7F2E 0A00                   DW 10
7F30 0100                   DW 1
7F30             ;--------------------------
7F32 010000      MATHS2     LD BC,0
7F35 7E          DH1        LD A,(HL)
7F36 FE30                   CP "0"
7F38 3838                   JR C DONE
7F3A FE3A                   CP "9"+1
7F3C 3034                   JR NC DONE
7F3E D630                   SUB 30H
7F40 FE0A                   CP 10
7F42 3809                   JR C DH2
7F44 D611                   SUB 11H
7F46 3821                   JR C BAD
7F48 FE06                   CP 6
7F4A 3F                     CCF
7F4B 381C                   JR C BAD
7F4D EB          DH2        EX DE,HL
7F4E 6F                     LD L,A
7F4F 2600                   LD H,0
7F51 09                     ADD HL,BC
7F52 3815                   JR C BAD
7F54 13                     INC DE
7F55 1A                     LD A,(DE)
7F56 FE30                   CP 30H
7F58 380A                   JR C DH4
7F5A 44                     LD B,H
7F5B 4D                     LD C,L
7F5C 3E09                   LD A,9
7F5E 09          DH3        ADD HL,BC
7F5F 3808                   JR C BAD
7F61 3D                     DEC A
7F62 20FA                   JR NZ DH3
7F64 44          DH4        LD B,H
7F65 4D                     LD C,L
7F66 EB                     EX DE,HL
7F67 18CC                   JR DH1

7F67             ;--------------------------
7F67             ;End of maths routines
7F67             ;--------------------------
7F67
7F69 E1          BAD        POP HL
7F6A C9                     RET
7F6A
7F6B E5          DECHEX     PUSH HL
7F6C 3620                   LD (HL),20H
7F6E 23                     INC HL
7F6F C3327F                 JP MATHS2
7F6F
7F72 E1          DONE       POP HL
7F73 23                     INC HL
7F74 78                     LD A,B
7F75 CD7F7F                 CALL BINHEX
7F78 79                     LD A,C
7F79 CD7F7F                 CALL BINHEX
7F7C 3648                   LD (HL),"H"
7F7E C9                     RET
7F7E
7F7F 57          BINHEX     LD D,A
7F80 0F                     RRCA
7F81 0F                     RRCA
7F82 0F                     RRCA
7F83 0F                     RRCA
7F84 CD887F                 CALL HEXOUT
7F87 7A                     LD A,D
7F88 E60F        HEXOUT     AND 0FH
7F8A C630                   ADD A,"0"
7F8C FE3A                   CP "9"+1
7F8E 3802                   JR C X1
7F90 C607                   ADD A,7
7F92 77          X1         LD (HL),A
7F93 23                     INC HL
7F94 C9                     RET
7F94
7F95 E5          HEXDEC     PUSH HL
7F96 3620                   LD (HL),20H
7F98 11227F                 LD DE,ST1-1
7F9B 0604                   LD B,4
7F9D 23          LP3        INC HL
7F9E 13                     INC DE
7F9F 7E                     LD A,(HL)
7FA0 FE30                   CP "0"
7FA2 3808                   JR C NOTNUM
7FA4 FE3A                   CP "9"+1
7FA6 3004                   JR NC NOTNUM
7FA8 D630                   SUB 30H
7FAA 180A                   JR VALID
7FAC FE41        NOTNUM     CP "A"
7FAE 38B9                   JR C BAD
7FB0 FE47                   CP "F"+1
7FB2 30B5                   JR NC BAD
7FB4 D637                   SUB 37H
7FB6 12          VALID      LD (DE),A
7FB7 10E4                   DJNZ LP3
7FB9 3A237F                 LD A,(ST1)
7FBC 17                     RLA
7FBD 17                     RLA
7FBE 17                     RLA
7FBF 17                     RLA
7FC0 E6F0                   AND 0F0H
7FC2 67                     LD H,A
7FC3 3A247F                 LD A,(ST2)
7FC6 B4                     OR H

7FC7 67                     LD H,A
7FC8 3A257F                 LD A,(ST3)
7FCB 17                     RLA
7FCC 17                     RLA
7FCD 17                     RLA
7FCE 17                     RLA
7FCF E6F0                   AND 0F0H
7FD1 6F                     LD L,A
7FD2 3A267F                 LD A,(ST4)
7FD5 B5                     OR L
7FD6 6F                     LD L,A
7FD6                        ;Binary now in HL
7FD7 D1                     POP DE
7FD7                        ;DE now points
                          into text file, at
7FD7                        ;the SYMbol
                            character
7FD8 C3007F                 JP MATHS1
7FD8
7FDB F5          CONVERT    PUSH AF
7FDC E5                     PUSH HL
7FDD D5                     PUSH DE
7FDE C5                     PUSH BC
7FDF 2A2388                 LD HL,(CURSOR)
7FE2 23          LP22       INC HL
7FE3 7E                     LD A,(HL)
7FE4 FE23                   CP SYM0
7FE6 CC957F                 CALL Z HEXDEC
7FE9 FE5E                   CP SYM1
7FEB CC6B7F                 CALL Z DECHEX
7FEE FE40                   CP SYM2
7FF0 2804                   JR Z EXIT11
7FF2 FEFF                   CP SYM3
7FF4 20EC                   JR NZ LP22
7FF6 C1          EXIT11     POP BC
7FF7 D1                     POP DE
7FF8 E1                     POP HL
7FF9 F1                     POP AF
7FFA C9                     RET
7FFA
7FFB             FIN2       EQU $
00FB             LEN2       EQU FIN2-START
00FB
00FB
```

symbol serves admirably as the end of record marker. **Listing 4** shows how the record would be typed into the wordprocessor, using the percentage character as the delimiting symbol.

Trailing spaces can be left out since the sorting routine can cope with variable field lengths. At print-time, the delimiting symbol is replaced by one or more spaces. It should, however, remain in the wordprocessor text until the user is satisfied that all sorting has finished, whereupon it can be removed by a normal global-replace instruction.

LISTING 2.

```
The text as written...
"address 32344 has code 005E."

Mark the decimal number...
"address ^32344 has code 005E."

Mark the hex number...
"address ^32344 has code #005E."

Invoke the conversion, giving...
"address  7E58H has code 00094."
```

Depth of sort

A film list of a few hundred titles can be sorted efficiently just by looking at the first character of field number three, the title field. The sort wouldn't be perfect, but who would notice if AIRPORT '80 was listed before AIRPORT '70? To untangle these two requires a sort which searches to a depth of 10 characters and thus distinguishes the "8" from the "7". Certain unsavoury films have very similar titles and sorting depths of 20 characters or more are needed to get them into their correct order. The rule is to sort to the minimum depth necessary for the task in hand. Every extra character taken into consideration means extra time on the sort, and it's quite possible to end up with a sort which takes 30 minutes or more if you're dealing with a thousand records to a depth of 20 characters, and that's in machine code!

LISTING 3.

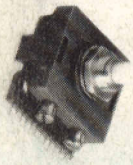
```
Stock Date Medium Title    Category Rating Comment
-----------------------------------------------------

1234 820714 VB BANANAS              C U Woody Allen
  12 810103 P  CARAVAN TO VACCARES  T X C.Rampling
 657 820622 VB CAT BALLOU           W U Jane Fonda
0123 831221 V  RETURN OF THE JEDI   S U Carrie Fisher
-----------------------------------------------------
```

LISTING 4.

```
<CR>0123%831221%V%RETURN OF THE JEDI%S%U%Carrie Fisher<CR>
```

More This Month at Maplin

256K D–RAM 41256 – 150ns ONLY £9.95 (QY74R).
256K EPROM 27256 – 250ns ONLY £18.95 (QY75S).
Right-angle pcb mounting rotary switches: 1P12W – FT56L; 2P6W – FT57M; 3P4W – FT58N; 4P3W – FT59P. All £2.95 each.
Stepper motor 48 steps/rev, 12V 0.13A per phase, 4-phase unipolar, 57g, working torque 8mNm max. ONLY £9.95 (FT73Q).
Driver*chip for motor: SAA1027 ONLY £3.75 (QY76H).
★SAVE★ 1 Kit containing everything you need: motor, SAA1027, data sheet and passives ONLY £13.35 (LK76H).



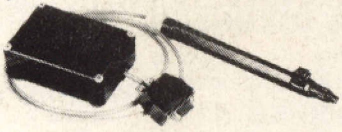
Sounds Terrific

Professional Quality
High Power Loudspeakers
featuring:

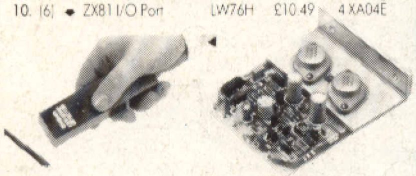
★ Virtually indestructible high-temperature voice-coil reinforced with glass-fibre.
★ 100% heat overload tolerance.
★ Advanced technology magnet system.
★ Rigid cast alloy chassis.
★ Linen or Plastiflex elastomer surrounds.
★ 5-year guarantee (in addition to statutory rights).

Prices from £17.97.
Send S.A.E. for our free leaflet XH62S.

Top Ten Kits

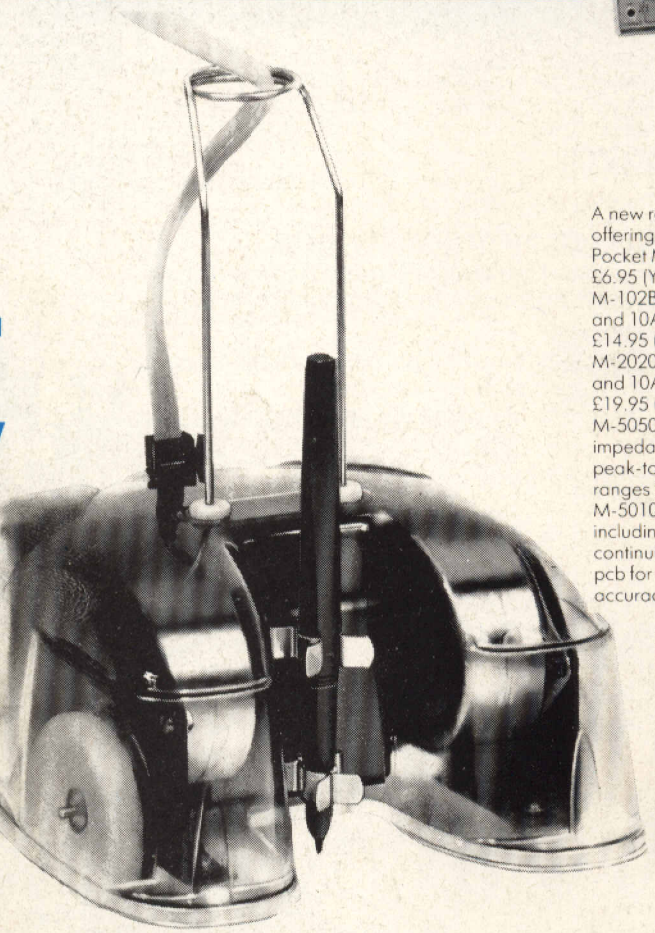


THIS/LAST MONTH	DESCRIPTION	CODE	PRICE	BOOK
1. (–)	◆ Live-Wire Detector	LK63T	£2.95	14 XA14Q
2. (1)	◆ 75W Mosfet Amp.	LW51F	£15.95	Best E&MM
3. (2)	◆ Partylite	LW93B	£10.95	Best E&MM
4. (4)	◆◆ Car Burglar Alarm	LW78K	£7.49	4 XA04E
5. (9)	◆ U/sonic Intrudr Dtctr	LW83E	£10.95	4 XA04E
6. (10)	◆ Computadrum	LK52G	£9.95	12 XA12N
7. (8)	◆ Light Pen	LK51F	£10.95	12 XA12N
8. (11)	◆ Syntom Drum Synth.	LW86T	£12.95	Best E&MM
9. (7)	◆ 8W Amplifier	LW36P	£4.95	Catalogue
10. (6)	◆ ZX81 I/O Port	LW76H	£10.49	4 XA04E



Over 100 other kits also available. All kits supplied with instructions. The descriptions above are necessarily short. Please ensure you know exactly what the kit is and what it comprises before ordering, by checking the appropriate Project Book mentioned in the list above.

Is it a turtle? Is it a robot? Is it a buggy? Yes! it's Zero 2.



- May be used by any computer with RS232 facility.
- Stepper Motor controlled.
- Half millimetre/half degree resolution.
- Uses ordinary felt-tip pens.
- Built-in 2-tone horn, line-follower. LED indicators.

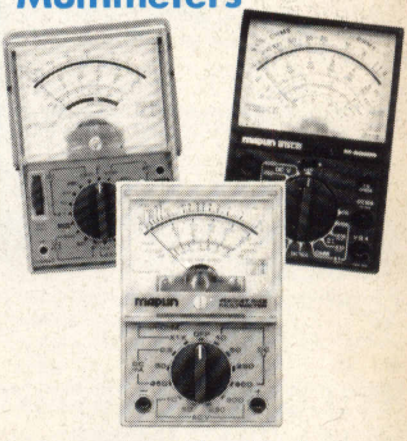
The Zero 2 Robot is the first truly micro robotic system available and remarkably it costs less than £80. Complete kit (only mechanical construction required) £79.95 (LK66W).
Full details of power supply and simple interfacing for BBC, Commodore 64 and Spectrum, in Maplin Magazine 15 price 75p (XA15R).

MAPLIN

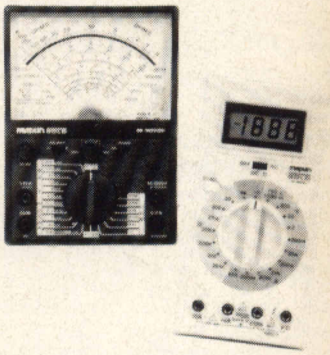
MAPLIN ELECTRONIC SUPPLIES LTD

Mail-order: P.O. Box 3, Rayleigh, Essex SS6 8LR.
Telephone: Southend (0702) 552911
SHOPS
• BIRMINGHAM Lynton Square, Perry Barr, Tel: 021-356-7292.
• LONDON 159-161 King Street, Hammersmith, W6.
Telephone: 01-748 0926.
• MANCHESTER 8 Oxford Road, Tel: 061-236 0281.
• SOUTHAMPTON 46-48 Bevois Valley Road. Tel: 0703-225831
• SOUTHEND 282-284 London Rd, Westcliff-on-Sea, Essex.
Telephone: 0702-554000. *Shops closed all day Monday.*

More Choice In Multimeters



A new range of very high quality multimeters offering truly amazing quality at the price.
Pocket Multimeter, 16 ranges, 2,000Ω/V DC/AC £6.95 (YJ06G)
M-102BZ with continuity buzzer, battery tester and 10A DC range, 23 ranges, 20,000Ω/V DC £14.95 (YJ07H)
M-2020S with transistor, diode and LED tester and 10A DC range, 27 ranges, 20,000Ω/V DC £19.95 (YJ08J)
M-5050E Electronic Multimeter with very high impedance FET input, 53 ranges, including peak-to-peak AC, centre-zero and 12A AC/DC ranges £34.95 (YJ09K)
M-5010 Digital Multimeter with 31 ranges including 20Ω and 20μA DC/AC FSD ranges, continuity buzzer, diode test, and gold-plated pcb for long-term reliability and consistent high accuracy (0.25% +1 digit DCV) £42.50 (YJ10L)



The Maplin Service

All in-stock goods despatched same day for all orders received before 2.00 pm.
All our prices include VAT and carriage (first class up to 750g).
A 50p handling charge must be added if your total order is less than £5.00 on mail-order (except catalogue).

AMERICAN EXPRESS · map · Access · VISA

☎ Phone before 2.00 p.m. for same day despatch.

1985
CATALOGUE

Pick up a copy now at any branch of W.H. Smith or in one of our shops. Price £1.35, or by post £1.75 from our Rayleigh address (quote CA02C).

All offers subject to availability.
Prices firm until 10th August 1985.